STUDY OF COMPUTATIONAL INTELLIGENCE ALGORITHMS TO DETECT BEHAVIOUR PATTERNS

Author: Fernando Palero Molina
Advisor: David Camacho Fernández

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

MÁSTER UNIVERSITARIO EN INVESTIGACIÓN E INNOVACIÓN EN TIC (i2-TIC)

TRABAJO FIN DE MÁSTER

October 2014

Para recorrer con éxito el camino,una mente abierta es la clave. GRACIAS por este camino. Por que de él aprendí: gratitud y humildad. Lo cual, me deja ver que lo natural es hermoso, honesto y brutal. GRACIAS!

Acknowledgements

This work has been supported by Airbus Defence & Space (Savier Project: FUAM-076914), and partially by the Spanish Ministry of Science and Education under the Project Code TIN2010-19872. I would like to express my grateful to the Savier project team specially I would like to mention to José Insenser, Juan Antonio Henriquez and Gemma Blasco.

Abstract [1]

In order to achieve the game flow and increase player retention, it is important that games difficulty matches player skills. As a consequence, to evaluate how people play a game is a crucial component, because detecting gamers strategies in video-games, it is possible to fix the game difficulty. The main problem to detect the strategies is whether attributes selected to define the strategies correctly detect the actions of the player. To study the player strategies, we will use a Real Time Stategy (RTS) game. In a RTS the players make use of units and structures to secure areas of a map and/or destroy the opponents resources. In this work, we will extract the real-time information about the players strategies using a platform base on the RTS game. After gathering information, the attributes that define the player strategies are evaluated using unsupervised learning algorithm (K-Means and Spectral Clustering). Finally, we will study the similitude among several gameplays where players use different strategies.

[1]This work has been funded by Airbus Defence & Space (Savier Project: FUAM-076914), and partially by TIN2010-19872.

Resumen[2]

A fin de lograr que el flujo del juego mejore y la captación de jugadores aumente, es importante que la dificultad del juego se ajuste a las habilidades del jugador. Como consecuencia, evaluar como juega la gente un juego es un aspecto importante, porque detectando las estrategias de los jugadores en los vídeo juegos, permite adapta la dificultad del juego. El problema principal para detectar las estrategias es si los atributos seleccionados para definir las estrategias definen correctamente las acciones del jugador. Para estudiar las estrategias de los jugadores, usaremos un juego de estrategia en tiempo real (Reat Time Strategy (RTS) en inglés). En un RTS los jugadores hacen uso de unidades y estructuras para asegurar áreas del mapa y/o destruir los recursos de los oponentes. En este trabajo, extraeremos información en tiempo real acerca de las estrategias usando una plataforma basada en un juego de RTS. Después de recoger la información, los atributos que definen las estrategias de los jugadores son evaluados mediante algoritmos de aprendizaje no supervisado (K-Means y Spectral Clustering). Finalmente, estudiaremos la similitud entre diversas partidas donde los jugadores utilizar diferentes estrategias.

[2]Este trabajo ha sido financiado por Airbus Defence & Space (Proyecto Savier: FUAM-076914) y parcialmente por TIN2010-19872.

Contents

List of Figures

List of Tables

Abbreviation

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CIM	**C**omputational **I**ntelligence **M**odule
DM	**D**ata **M**aning
DRM	**D**ata **R**ecovery **M**odule
GD	**G**ame **D**ata
GS	**G**ame **S**napshot
ID	**I**nterplay **D**ata
KDD	**K**nokledge **D**iscovery **D**atabases
k-NN	**N**earest **N**eighbour method
MVM	**M**odel **V**alidation **M**odule
NPC	**N**on-**P**layer **D**haracter
OTD	**O**pen-source **T**ower **D**efence
PI	**P**layer **I**nteraction
PTSP	**P**hysical **T**ravelling **S**alesman **P**roblem
WGM	**W**ave **G**enerator **M**odul

Chapter 1

Introduction

Nowadays a wide number of Computer Science researchers are focused on intelligence Video Games [66, 97, 110, 114, 116], design and development. Several techniques and methods from areas such as Artificial Intelligence (*AI*) or Data Mining (*DM*) have been applied to analize the player behaviours analysis [74, 108], intelligent enemies [116, 121], or to imitate the human behaviour [88, 102]. One of the most popular applications is related to the development of controllers to automatically define real behaviour of Non-Player Characters (*NPC*). In this topic, there are several works focused on really famous games such as *Ms. PacMan* [81], *Physical Travelling Salesman Problem* (*PTSP*) [103], *Super Mario Bros.* [90] or *Starcraft* [115]. Other works have been focused on the generation of automatic levels [111] or the validation of those levels by finding the different paths that reaches the exit [83, 84].

This work presents an initial case study related to the identification and analysis of behaviours in Video Games using as a first approach visualization techniques to understand what happened during the game execution. For example, the *ESOM* [77] visualization technique has been used to identify the groups of player patterns in the famous game *Tomb Raider Underworkd* [77]. Other example can be found in [76] where authors study the user behaviour in the game *Kane & Lynch and Fragile Alliance* by the use of diagrams to understand the reasons of why the players die and map levels visualizations to represent the paths followed by the players. After the visual analysis to identify the different player strategies, clustering algorithms (K-Means and Clustering) and an evaluation function will be employed to select the set of attributes which correctly detect the players actions.

The platform designed to achieve our goals, to study the players behaviour and to select the game features that best define the behaviour, is based on an open-source *Tower Defence*

game[1], called *OTD*, that it is a subgenre of real-time strategy Video Game. This analysis is based on previous works [68, 85] where the interaction of users were automatically extracted and analyzed from an Educational Virtual World to differentiate the different student communities based on the interactions among them.

The goal in *OTD* game is to avoid that a set of enemies, reaches the exit of the level. In order to do that, players need to build in the map different traps, or some defensive buildings, that difficult enemies to cross the map. In this work, the players behaviour is analysed by taking into account only the position of the different towers placed by the user in a 33×60 grid.

Two different kind of experiments have been carried out in this work. In the first one, Visualization techniques have been applied to identify the player strategies, in our first experiment the players have been divided in two groups. The first one was formed by players with an specific strategy assigned and the second one, the players played without a prefixed strategy. Then, the different gameplays were analysed using visualization techniques. After the analysis we conclude: on one hand, the attributes employed to identify the strategies are not enough it is necessary to study new features and, on the other hand, the numbers of strategies employed for the players was reduce to 3.

The second experiment was focus on studying new game features, for this purpose clustering algorithms were used with an evaluation function to choose the group of attributes which best fit the player strategies. Experimental results reveal that the set of attributes that best define the player strategies are the X, Y coordinates, which define the position of the towers.

[1] Available at: http://sourceforge.net/projects/otd/

Chapter 2

State of the Art

2.1 Pattern Recognition

Pattern detection is nearly synonymous with machine learning [67]. This branch of AI focuses on the recognition of patterns and regularities in data. In many cases, these patterns are learned from labelled "training" data (supervised learning), but when no labelled data are available other algorithms can be used to discover previously unknown patterns (unsupervised learning).

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. The common supervised methods are:

- Support Vector Machines [87] are a group of supervised learning methods that can be applied to classification or regression. Support vector machines represent an extension to non-linear models of the generalized portrait algorithm developed by Vladimir Vapnik.

- Linear regression [72] attempts to model the relationship between two variables by fitting a linear equation to observed data.

- The Naive Bayes [106] technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

- An Artificial Neural Network (ANN) [91] is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems.

- Nearest neighbour method [86] (k-NN) is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

- Decision tree [109] builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Unsupervised learning aims to organize a collection of data items into clusters, such that items within a cluster are more "similar" to each other than they are to items in the other clusters. This notion of similarity can be expressed in very different ways, according to the purpose of the study, to domain-specific assumptions and to prior knowledge of the problem. Unsupervised learning is usually performed when no information is available concerning the membership of data items to predefined classes. The methodologies employed to unsupervised learning are:

- Clustering is the task of grouping a set of objects in such a way that objects in the same group have similar characteristics to each other more than in other groups. Example of this algorithm are K-Means [122], Mixture Models [80], Hierarchical Clustering [89], etc.

- Hidden Markov model is a tool for representing probability distributions over sequences of observations. They are used in almost current speech recognition systems [104], in numerous applications in computational molecular biology [92], in data compression [69], etc.

The terms pattern recognition, DM [107] and knowledge discovery in databases [79] (KDD) are hard to separate, as they largely overlap in their scope. Machine learning is the common term for supervised learning methods and originates from AI, whereas KDD and DM have a larger focus on unsupervised methods and stronger connection to business

use. Pattern recognition has its origins in engineering, and the term is popular in the context of computer vision. In pattern recognition, there may be a higher interest to formalize, explain and visualize the pattern; whereas machine learning traditionally focuses on maximizing the recognition rates. Yet, all of these domains have evolved substantially from their roots in AI, engineering and statistics; and have become increasingly similar by integrating developments and ideas from each other.

In machine learning, pattern recognition is the assignment of a label to a given input value. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes (for example, determine whether a given email is "spam" or "non-spam" [93]). However, pattern recognition is a more general problem that encompasses other types of output as well. Other examples are regression, which assigns a real-valued output to each input; sequence labelling [75], which assigns a class to each member of a sequence of values (for example, part of speech tagging [82], which assigns a part of speech to each word in an input sentence); and parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence.

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to perform "most likely" matching of the inputs, taking into account their statistical variation. This is opposed to pattern matching algorithms [78], which look for exact matches in the input with pre-existing patterns. A common example of a pattern-matching algorithm is regular expression matching [99], which looks for patterns of a given sort in textual data and is included in the search capabilities of many text editors and word processors. In contrast to pattern recognition, pattern matching is generally not considered a type of machine learning, although pattern-matching algorithms can sometimes succeed in providing similar-quality output to the sort provided by pattern-recognition algorithms.

Pattern recognition is studied in many fields, including psychology, psychiatry, ethology, cognitive science, traffic flow and computer science.

This research has been focused on computer science discipline. Concretely, as previously mentioned, pattern recognition. To find the patterns we use clustering methods, because we try to identify humans behaviour but we do not know beforehand which are these behaviours. Then, clustering groups the data and using these groups we can generate the patterns necessaries to identify the humans behaviour.

Pattern recognition is employed in a range of computing fields where designing and programming explicit, rule-based algorithms is infeasible. Example of fields are:

- AI is an area of computer science that deals with giving machines the ability to seem like they have human intelligence [73].

- Information retrieval is the task of finding automatically in a large corpus the documents that are relevant to an information need expressed through a query [113].
- DM is the set of techniques and technologies that allows to explore big databases, using automatic or semi-automatic methods, with the aim to find repetitive patterns, trends or rules which define the behaviour of the data in a specific context.
- Human behaviour Recognition is the area focused on the fields human computer interaction [98], affective computing [101] and social signal processing [117]. The aim of this fields is to do easy the communication human-machine.

2.2 Human behaviour Recognition in Video games

In this work we are interested on Human behaviour Recognition field. Human behaviour detection is a crucial need rely on advanced pattern recognition techniques to automatically interpret complex behavioural patterns generated when humans interact with machines or with others. A test-bed to study methods of human behaviour recognition are the video games. The reason of this is the huge datasets available for analysis resulting from players engaging in interactive environments. These datasets enable investigation of individual player behaviour at a massive scale.

Traditionally the utilization of techniques from the AI and Human Behaviour Recognition (HBR) areas have been employed in classical board games like chess, checkers, kalaha, go, othello, Tic-Tac Toe, etc... [49, 51]. However, the high impact of video game industry has originated an increasing interest in the practical utilization of AI and HBR techniques in commercial video games. Currently, there exist a wide number of international conferences that propose different competitions, or challenges, whose main goal is based on the utilization of AI and HBR techniques to solve problems like: automatic generation content to match the game levels with the player skills; definition of autonomous Non-Player Characters that simulates human behaviour (i.e. autonomous bots); learning; etc...

Maybe, the most popular competitions are those from the IEEE International Conference on Computational Intelligence and Games (www.ieee-cig.org), this conference has promoted different competitions in the last ten years based on popular video games like Mario Bros, Ms-Pacman or StarCraft. The basic idea behind these competitions is based on the utilization of a video game platform that can be used to integrate AI/HBR algorithms to test their behaviour in a particular challenge.

2.2.1 Current Video games Platforms & Competititons

This section will briefly describe the main, and most popular, video games competitions offered by CIG conference. In the last edition (CIG'2013) the set of competitions proposed was: Student Video Competition; Physical Travelling Salesman Problem (PTSP); Multiobjective - Physical Travelling Salesman Problem (MO-PTSP); Geometry Friends; Platformer AI Competition (formerly Mario AI); StarCraft RTS AI Competition. From previous competitions, only some of them directly related to our work (identify behaviour patterns) will be analysed.

Ms Pac-Man

Ms Pac-Man is a predator-prey arcade game, where the species, pac-man and ghost, compete, evolve and disperse simply for the purpose of seeking resources to sustain their struggle for their very existence. The game consists of a maze with paths and corridors that the Pac-Man moves through collecting food pills that fill some of these paths. The aim of the game is to control the Pac-Man in order to clear all the pills in the current maze and then advance to the next one. To do that, the methodologies based on AI/HBR techniques that have been used for that game are mainly based on Genetic Programing and Coevolution [50].

During the game, the Pac-Man is chased by four ghosts any of whom will kill the Pac-Man if they are able to catch him. The ghosts behave in a non-deterministic way, which makes it hard to predict their next move although their general behaviour varies from random to very aggressive. The goal of a **Ms Pac-Man controller** is to maximise the score of the game. In the competition, it is the average score against multiple ghost teams that counts and the winning controller is the one which obtains the highest total average score. Therefore, the goal of a *ghost-team controller* is to minimise the average score obtained against it by the different Ms Pac-Man controllers. The winning ghost team will be the team with the lowest average score against it.

Platformer AI

The Platformer AI Competition is the successor for the Mario AI Championship (www.marioai.org), and it is focused on two main AI topics: procedural content generation (i.e. level generations) and imitating human behaviour. In the past IEEE-CIG 2012 competition, the Turing Test Track from the Mario AI Championship was focused on developing *human-like con-*

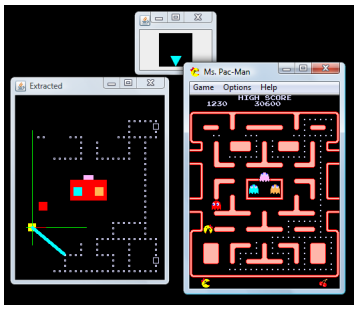

Fig. 2.1 Ms Pac-Man Video Game Platform

trollers [100].

The test bed game used for the competition is a modified version of *Markus Persson Infinite Mario Bros* (mojang.com/notch/mario/) which is a public domain clone of Nintendo classical two-dimensional platform game Super Mario Bros. The game-play in *Super Infinite Mario Bros* takes place on two-dimensional levels in which the player (Mario) has to move from left to right avoiding obstacles and interacting with game objects. Mario can move left, right and duck, additionally two keys can be used to allow Mario to run, jump, or fire (depending on the state he could be in the game).

One of the main goals of this competition is to be able to compare different controllers development methodologies against each other, these controllers can be based on different AI/HBR techniques such as artificial evolution, evolutionary neural networks, genetic programming, fuzzy logic, temporal difference learning, human ingenuity, hybrids of the above, etc [61].

Physical Traveller Salesman Problem

The Physical Travelling Salesman Problem (PTSP) is a real-time game played on a two-dimensional map. The map has walls and obstacles, and several waypoints. In the Multi Objective PTSP competition each map has 10 waypoints. The map itself is represented as a bitmap, where each pixel is either a wall or an empty space. The game proceeds in discrete time steps, usually one every 40 ms (i.e. 25 per second). The player controls a spaceship, similar to the one in the classic video game **Asteroids**. The ship can rotate using a constant angular speed, and can apply thrust in the direction that it is currently pointing. The goal is to minimize three different objectives: time taken to complete the maze, fuel consumed in

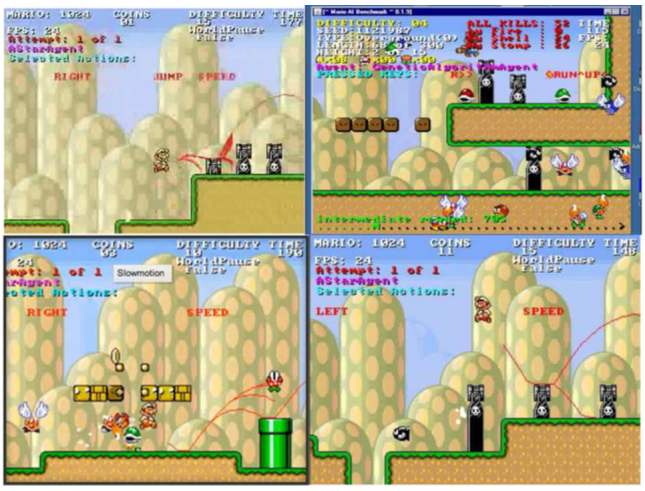

Fig. 2.2 Mario AI Video Game Platform

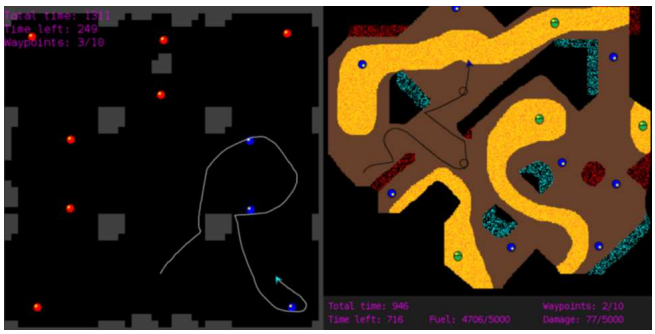

Fig. 2.3 TSP Video Game Platform

the process, and damage taken by the ship.

In this challenge like in Platformer AI, the aim is to be able to compare different controllers development methodologies against each other, such as Monte Carlo tree search, evolutionary neural networks, niched pareto genetic algorithms, non-dominated sorting genetic algorithms, strength pareto genetic algorithms, etc [20].

StarCraft

Blizzard's StarCraft is one the most popular, and fun, examples of the real-time strategy (RTS) genre. In this game, a set of races (Protoss, Zerg and Terrans) can be used to build units that have access to different technology skills, every unit works differently and requires different tactics for a player to succeed.

The enigmatic Protoss have access to powerful units and machinery and advanced technologies such as energy shields and localized warp capabilities, powered by their psionic traits. However, their forces have lengthy and expensive manufacturing processes, encouraging players to follow a strategy of the quality of their units over the quantity. The insectoid Zerg possess entirely organic units and structures, which can be produced quickly and at a far cheaper cost to resources, but are accordingly weaker, relying on sheer numbers and speed to overwhelm enemies. The Terrans provide a middle ground between the other two races, providing units that are versatile and flexible. They have access to a range of more ballistic military technologies and machinery, such as tanks and nuclear weapons.

There are two different game modes; one against one, and teams. The game's goal is to compete for resources and destroy the enemy. For this reason, once the game has started, the players must recollect raw material and build as quick as possible, factories, buildings, etc. During the game, both players (or teams) are constantly evolving to overcome the opponent, winning land and destroying enemies settlements. To do this it is needed that the player continuously evolves and adapts the strategy in function to the enemy movements.

The CIG StarCraft[1] competitions have shown some relevant advances in the development and evolution of new StarCraft bots. Although human top Starcraft players remain unbeaten, the machines are striving to close the gap between human and AI.

Although each race is unique in its composition, no race has an innate advantage over the other. Each specie is balanced out, so they have different strengths, powers, and abilities. But their overall strength is the same and therefore an ideal candidate to test different AI approaches like: neural networks, evolutionary algorithms, fuzzy systems, swarm intelligence and artificial immune systems[14].

2.2.2 Summary on Video Games Platforms for AI/HBR

Finally, Table 2.1 summaries the algorithms and the applications of AI/HBR in video games. In this table, the following information is shown:

1. **Video Game**. Examples of video games where player behaviour recognition have

[1] ls11-www.cs.uni-dortmund.de/rts-competition/starcraft-cig2013

Fig. 2.4 Starcraft Video Game Platform

been used to some purpose.

2. **Algorithms & Method**. The algorithm and methods employed in each paper to study the player behaviour.

3. **Goals**. The main goals that must be reached for each paper.

4. **Results**. The conclusions and results obtained when the different methods and algorithms have been applied.

Although a wide number of algorithms and techniques from CI (i.e. Neural Networks, Evolutionary Strategies or Fuzzy Logic) and other AI methods (from heuristic search to statistical methods) have been successfully applied, the Lemmings video game provides two new interesting features. On the one hand, the video game provides different kind of terrains, that the algorithm must take into account to avoid a premature dead of the lemming, or to decide an adequate selection from the available skills. This characteristic provides an interesting ”context” that should be handled by the algorithm (for instance, by using a constraint-based modelling of the environment or a meta-heuristic to select the best skill). On the other hand, the game itself needs from the management and control of a *colony* of Lemmings. It is necessary to coordinate those lemmings to look for the best solution. However, the optimum solution is base on a mixture of different goals, so multi-objective algorithms can be easily applied in this domain.

Video Game	Algorithms & Method	Goals	Results
Pac-Man [70]	**Algorithm**: Genetic Programming (GP) **Method**: The algorithm uses data extracted (Gost position, pills position, ms Pac-man position, free path, the current number of pills in the game, etc) from games played by humans to train the GP algorithm. Also, during using information retrieval methods obtaion the game the game state. Then, with the current game state information the GP algorithm is evolved to find the best solution. The fitness function used to choose the best solution calculate average game score. The score is calculated in function of the number of pills that ms Pac-Man ate.	To create a non-player character to control ms Pac-Man	The experimental data shown that the same controller may achieve results from about 5,000 to 35,000 points in two consecutive simulation runs. This happens because the various sample games played by human players during the experiments have expert and novice human players. Then, the game controller may also achieve bad results in some sample games.
Pac-Man [120]	**Algorithm**: Neuro-evolutionary on-line learning. **Method**: In this research has been design a Player Modelling (PM) using an evaluation function to identify the impact on the interest of the game. To do this, the model is combined with the Neuro-evolutionary on-line learning procedure. More specifically, the model, Bayesian Networks (BN) model, is trained on computer-guided player data, as a tool for inferring appropriate parameter values which generate games of higher interest for the player. The evaluation function employed are given by T (challenge metric; based on the difference between maximum and average lifetime of the players over N games — N is 50 in this paper), S (diversity metric; based on standard deviation of lifetime of the players over N games) and EH_n (aggressiveness metric; based on stage grid-cell visit average entropy of the Ghosts over N games) respectively. All three metrics are combined linearly 2.1 where I is the interest value; α, β and γ are criterion weight parameters ($\alpha = 1$, $\beta = 2$, $\beta = 3$ for the experiment). $$I = \frac{\alpha T + \beta S + \gamma EH_n}{\alpha + \beta + \gamma} \quad (2.1)$$	To adapt the the levels difficulty to match with the player skills to increase the increase the interest of the players.	In the paper is demonstrated that PM mechanism developed has a positive impact on the generation of more interesting games. Moreover, the proposed PM-Online learning mechanism shows game reliability since it demonstrates adaptive behaviours in the scale of decades of games played and it is computationally inexpensive (1-3 seconds of CPU time for the BN to infer OLL parameter values).

Mario [112]	**Algorithm**: **Method**: The personalized levels are automatically generated for platform games using models which predict player experience based on features of level design: (a) Controllable features of the game: These parameters are used for level generation, and affect the type and difficulty of the level. This set contains three features that are related to gaps: number of gaps, average width of gaps, and gap entropy, as well as a switching feature that defines the percentage of the level played in the left direction. (b) Gameplay characteristics: Statistical features of how the user plays the game such as how often the player jumped, ran, died, how much he spent moving left, and how many enemies he killed for the different type of opponents. These features cannot be directly controlled by the game as they depend on the skill of the player and playing style. And (c) the player experience: After playing a set of four games, divided into two pairs played in both orders, players were asked to report the preferred game for three emotional dimensions; fun, challenge and frustration, through a 4-alternative forced choice questionnaire protocol. With all this feature a single layer perceptrons (SLPs) have been used to create a model to predict the affective state of the players. Sequential feature selection is used to choose the input subsets for the SLPs.	To generate automatic personalized content for Platform Games	The experimental results of this research show that using the proposed approach the model is able to generate levels tailored to specific players. Also, the model achieved acceptable accuracy, it is able to predict the emotional states with a relatively high precision and adapt well to the playing style of the players.
Starcraft [119]	**Algorithm**: **Method**: In this work use a DM approach to opponent modeling in strategy games. Expert gameplay is learned by applying machine learning techniques (Decision tree [], Nearest neighbour, Non-nested generalized exemplars [] and Additive logistic regression []) to large collections of game logs. This approach enables domain independent algorithms to acquire domain knowledge and perform opponent modelling. Machine learning algorithms are applied to the task of detecting the strategy of an opponent before it is executed and predicting when an opponent will perform strategic actions. The approach used to define the opponent strategy involves encoding game logs as a feature vector representation, where each feature describes when a unit or building type is first produced.	To create a non-player character enemy which can modify its strategy according human player behaviour.	The experimental result reveal, that the model proposed by the researchers in perfect and imperfect information environments the results show that their representation has higher predictive capabilities, and is more tolerant of noise.

Table 2.1 Algorithms and the applications of AI/HBR in video games

In the section 2.1 has been mentioned that this research has been focused on computer science discipline. The goal of this work is to study the human behaviour along the time. To do this, we decide to use a real time strategy game due to the huge datasets available for analysis the human behaviour, in this case we look for to identify player strategies. Then,

we want to use pattern recognition which allow us to identify the player strategies. Clustering methods will be used to find the patterns, because the goal is to identify humans behaviour but we do not know beforehand which are these strategies. Then, Using clustering techniques to groups the data, we can extract the patterns from these clusters (groups) necessaries to identify the player strategies. Furthermore, the data instances which from the groups will be vectors composed by the game attributes. It is necessary to study which set of attributes define better the player strategies, for this purpose, an evaluation function will be used to select the best combination of features.

Chapter 3

OTD Framework

3.1 Tower Defence Framework Architecture

This section presents a framework architecture based on a *OTD* platform (see Figure 3.1) that have been developed for us to study the players behaviours. The framework has been designed using four different modules. The Wave Generator Module (*WGM*) is the responsible to generate a fix number of variable hordes of enemies in each wave. The Data Recovery Module (*DRM*) allows to automatically extract data from the game platform, and gather the interaction from the users. The Computational Intelligence Module (*CIM*) analyses the state of the gameplay at the beginning of a new horde and return the suitable counter-strategy. The Model Validation Module (*MVM*) analyses and returns the distribution of the gameplays. With these distributions, visualization techniques (histograms) have been used to determine the strategies.

The framework is presented as a client-server model. The client is composed by the *WGM* and the *OTD* Game itself. The server provides a web service containing the *DRM* and *CIM* modules, which communicates with a database to store the information gathered and load the necessary information to analyse the user behaviour given the state of the board at the beginning of a new wave.

The *WGM* determines the size of the horde in a wave. The equation 3.1 is used to define the number of enemies of each type in the horde that will be generated in a gameplay. Where N represents the number of enemies, β_T the percentage of enemies of type T, W represents the current wave and α is a growing factor for the enemies generation. For $\alpha = 0$, the number of enemies in the horde is constant in each wave. If $\alpha = 1$, the amount of enemies grows

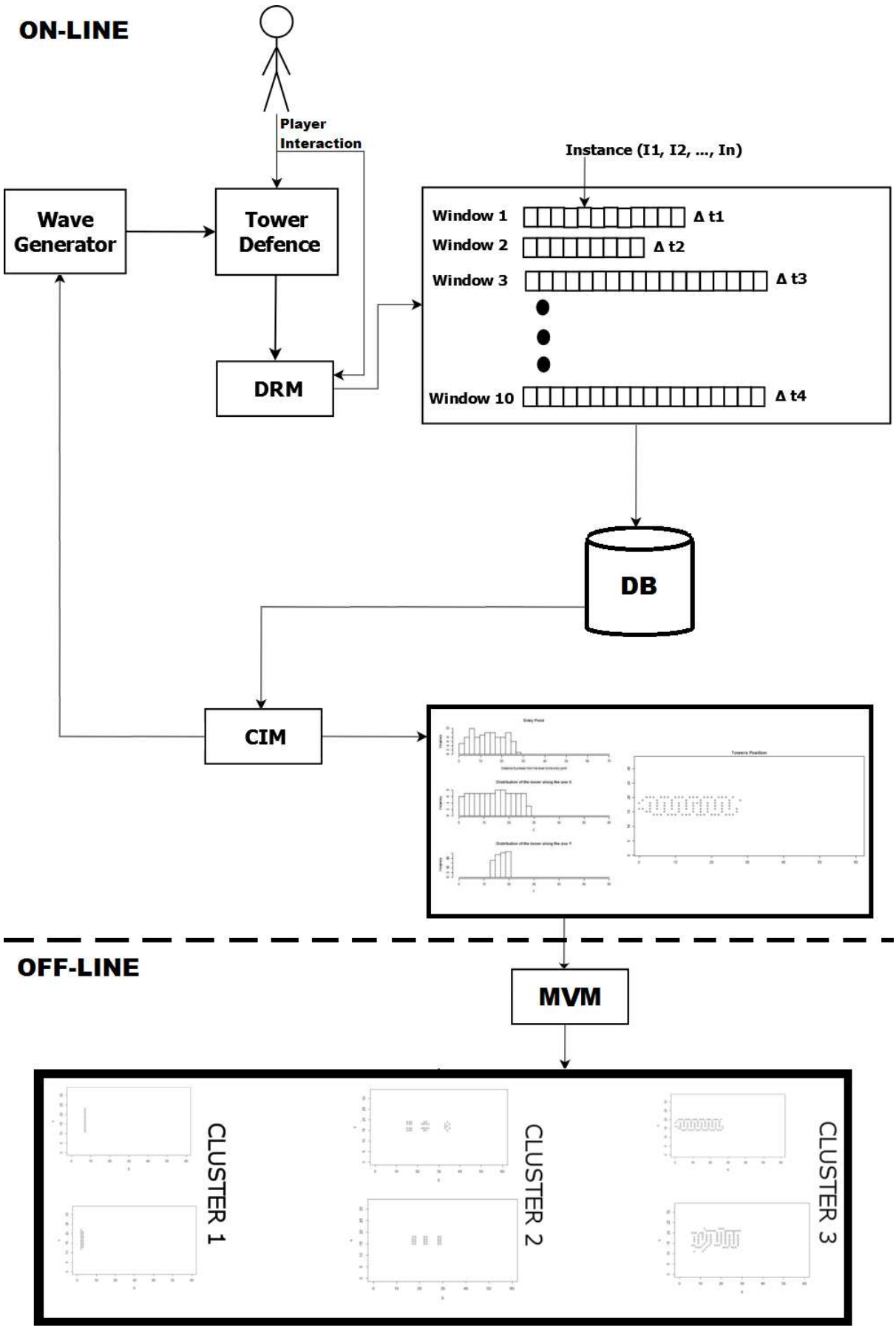

Fig. 3.1 Framework Architecture based on the *OTD* game platform

linearly and, finally, if $\alpha = 2$ the horde growth has quadratic grows factor.

$$\#Enemies_T = \beta_T NW^{\alpha} \tag{3.1}$$

The *DRM* extracts data from the Tower Defence game, which is pulled in two different categories: Game Data (*GD*) and Interplay Data (*ID*). *GD* provides information from the environment, which is based on the features predefined by the game (i.e. the number of waves in a game, the type of the enemies, the size of an horde, etc). On the other hand, there are two types of *ID*: Player Interaction (*PI*) and Game Snapshot (*GS*). *PI* data represents the actions of the player, and the events of the game during the execution, and *GS* providing a detailed description about state of the game for each instant of time. Both kind of data (*PI* and *GS*) are gathered from the platform every second during a gameplay. All this gathered information is stored into a database, and will be recovered by other modules to be analysed.

Finally, the *CIM* is the responsible to carry out the analysis of the data gathered by *DRM*. It works based on two basic processes: the first process recovers the data from the database, and the second process is used to compute the analysis. Using these features a initial classification of players strategies is generated.

On the other hand, *MVM*, which works off-line, is responsible for carrying out the analysis of the data gathered by DRM module for several gameplays. It works on two basic phases. In the first phase, each instance recorded in the data window is normalized and labelled, and the behaviour operators that help to identify the strategy are obtained. This label will be used to identify the cluster where the instances have been assigned. The algorithm used is based on K-means (see section 3.5.2). The second phase consists on studying the similitude between the gameplays strategies (see section 3.5.4).

3.2 OTD Specifications

To study the players behaviour an *OTD* game has been employed. For our purpose, it is necessary to customize the *OTD* game environment to create appropriate experiments. The next sections provide information about the *OTD* parameters.

3.2.1 Gameplay Parameters

Gameplay parameters define the level environment: the level difficulty, the duration of the level, the number of enemies, etc. When some parameter change, the players must adapt

their behaviour to the new situation. And these changes are analysed to extract pattern behaviour. The gameplay parameters are:

- Number of enemies: $[0-10M]$.
- Number of hordes: $[0-10M]$.
- Enemy spawning interval (Spawning = item creation, in this case the enemies): $[0-3,4*10^{38}]$ seconds.
- Time-out between waves: $[0-3,4*10^{38}]$ seconds.
- Number of initial player lives: $[1-2,3*10^{16}]$.
- Number of player lives: $[0-2,3*10^{16}]$
- Number of waves: $[1-2,3*10^{16}]$

3.2.2 Game Board Parameters

The game board is the place where all the actions happen. On one hand, the players place towers to destroy the enemies and, in the other hand, the enemies must find a path where the towers cannot defeat them. The size and types of tiles of the board is important because the configuration of the board alters the players and enemies behaviour. The game board parameters are:

- Grid size: $[(2\times2)-(1002\times1002)]$ tiles
- editable tiles: tiles where the player can put towers and where terrestrial enemies can go through. $[(2\times2)-(1002\times1002)]$
- non-editable tiles: tiles where the player can not put towers and where terrestrial enemies can not go through, only the aerial: $[(2\times2)-(1002\times1002)]$
  - If in the grid there are not ways of editable tiles between the start and the exit point, terrestrial enemies are not allowed. In this case, the game can only generate aerial enemies.
- Start points and Exit points
  - The grid has always at least one start and one exit point.

  - Start and Exit point can be placed anywhere in the grid.
  - Number of Start points: [1 – 1002003] tiles
  - Number of Exit points: [1 – 1002003] tiles

3.2.3 Resources Parameters

The number of initial resources and the amount of resources that an enemy throws condition the player behaviour. In this game resources are coins, which are necessary to buy towers. At the beginning of the game, the player start with a determinate amount of coins, this number of coins can vary between 5 and $3 * 10^{16}$ (5 is the minimum quantity to buy at least one tower). Furthermore, when defeats enemies, the number of resources that each enemy can proportionate are between $[0 – 3 * 10^{16}]$

3.2.4 Towers Parameters

The towers are the tools to defeat the enemies. There are 3 types of towers with different behaviours and features. The features can modify the behaviour of the towers to allow the player uses the towers in different situations. The parameters are:

- There 3 types of towers:
  - Canon: This tower shots bullets
  - Stunner: This tower slows down the enemies
  - RocketLauncher : This tower shots missiles
- The parameters of the towers are:
  - SellPrice: the number of resources that the player recovers when sells the tower. $[0 – 3,4 * 10^{38}]$ coins
  - Price: The number of resources that the player consumes to build a tower. $[0 – 3,4 * 10^{38}]$ coins
  - Shot: This parameter determines the time between shots
  - Distance: the scope of the tower shot (Radio). [1 – 1002] tiles
  - Power: This parameter determines the shot damage. $[0 – 3,4 * 10^{38}]$
  - Life: Life indicates the number of shots that the tower can receive. $[1 – 3 * 10^{16}]$

3.2.5 Enemies Parameters

There are 2 types of enemies with different behaviours and features. The possible behaviours are flight or walk. The features modify the lives of the enemies, speed, etc. The modification of the feature modify the difficulty in a gameplay. This is useful to study the player reactions. The enemies parameters are:

- There are two basic types of behaviours enemies:
  - Aerial: this kind of enemies can fly over the towers.
  - Terrestrials: this enemies can not go through the towers, for this reason they choose a path to avoid towers.
- For each basic types of enemies there are subtypes of enemies, where the difference between subtypes are the value of the parameters.
- Parameters:
  - Speed: the number of pixels walked per second. $[0-3,4*10^{38}]$
  - Shot: This parameter determines the time between shots. $[0-3,4*10^{38}]$
  - Power: This parameter determines shot damage. $[0-3,4*10^{38}]$
  - Life: Life indicates the number of shots that the tower can receive. $[0-3,4*10^{38}]$
  - Type: Aerial / Terrestrial.
  - Money: Number of coins received when the enemy is defeated. $[0-3*10^{16}]$ coins
  - Bonus: Number of Bonus points received when the enemy is defeated. $[0-3*10^{16}]$

3.2.6 Game Events

During a gameplay each time-step information about the game is stored but there are different events that it is necessary catch to study the players behaviours, these events are:

- To buy a tower.
- To sell a tower.

- To destroy enemies.
- To win extra life.
- To lose a life
- To earn resources.
- The apparition of a hordes.
- To win/lose the gameplay.
- When a tower is destroyed.

3.3 Data Extracted

To analyse the player behaviour 4 types of raw data has been extracted from *OTD* and stored in 7 different tables (this raw data is pre-processed to obtain the (*PI*) and (*GS*) data): level, game, enemies and towers data. Level data provides information about the characteristics of the level. Game data is the information related to the environment of the game, this means all the features predefined by the game, as has been aforementioned in Section 3.1. Finally, Enemies and Tower data show the state and status of the towers and enemies.

3.3.1 Table Level

The table Level stores the characteristics of the level. This is useful to related gameplays with the players and then study the player behaviour and to know the difficulty of the level. The table consists of 6 attributes.

- idLevel: this attribute identify a gameplay with a number.
- numLevel: numLevel indicates the mission played by the user.
- UserId: this parameter identify the player with a number.
- numEnemies, alpha, numWaves: these parameters correspond to the parameters of the equation 3.1.

3.3.2 Table Gamedata

This table store the information about the game status. This information allows to study the game features like the money, user health and wave number along the time. All these information is stored in the table each time-step.

- idLevel: this attribute relates the table GameData with the tables Level, Towers and Enemies.
- Health: Health determines the lives of the player in a specific time-step.
- Money: Money determines the resources of the player in a specific time-step.
- Wave: Wave gives information about the level stage.
- Step: this parameter gives information about the instance of time where the data has been stored in a gameplay.

3.3.3 Table Towers

This table stores the state and the type of the towers.

- idTower: this parameter identifies a tower.
- idLevel: this attribute relates the table Towers with GameData, Level and Enemies tables.
- Type: Type determines the tower skills.
- isDestroyed: this is a boolean value, this value shows if the tower has been destroyed.
- isRemoved: this is a boolean value, this value shows if the tower has been removed.

3.3.4 Table Towers_Status

The current status (life, level, power, etc) of the towers stored in the table Towers is saved in the table Towers_Status. The current status of the tower is stored each time-step.

- idTower: this attribute relates the table Towers_Status with the table Towers.
- Level: Level shows the level of the tower, each tower has 4 levels. The level determines the strength of the tower. The level 1 is the lowest and 4 is the highest.

- Shot: this parameter determines the time between shots
- Power: this parameter determine the shot damage.
- Distance: the scope of the tower shot.
- Energy: the life of the tower.
- Pos_X: the position *X* of the tower in the game board.
- Pos_Y: the position *Y* of the tower in the game board.
- Cluster: in the board can be several groups of towers, the parameter Cluster shows in which group appears the tower.
- Step: this parameter gives information about the instance of time where the data has been stored in the table Towers_Status.

3.3.5 Table Enemies

This table stores the state and the type of the enemies.

- idEnemy: this parameter identifies a enemy.
- idLevel: this attribute relates the table Enemies with the tables Level, Towers and GameData.
- Type: Type determines the enemy skills.
- Wave: Wave gives information about the moment in which the enemy appears.
- isDestroyed: this is a boolean value, this value shows if the enemy has been destroyed.

3.3.6 Table Enemies_Status

The status of the enemies stored in the table Enemies is saved in the table Enemies_Status. The current status of the enemy is stored each time-step.

- idEnemy: this attribute relates the table Enemy_Status with the table Enemy.
- Energy: the life of the enemy.
- Pos_X: the position *X* of the enemy in the game board.

- Pos_Y: the position *Y* of the enemy in the game board.
- Money: The number of resources gained when the enemy is defeated.
- Speed: the number of pixels walked per second.
- Step: this parameter gives information about the instance of time where the data has been stored in the table Enemies_Status.

3.4 Features Extracted

In order to define the player behaviours different strategies have been into in account. The goal of the experimental phase (see section 4.1) is to test whether the proposed framework is able to identify the different strategies. These strategies are defined taking into account the features: the distribution of tower position, the shortest path between Starting point and Exit point, number of turns that the enemy must perform to reach the exit point of the level and kill ratio of the users. The four different strategies studies are: Zigzag distribution, Vertical distribution, Horizontal distribution and Grouped distribution. These features will be utilized to define the player strategies. The features are:

3.4.1 Distribution of Tower Position

Using the tower positions, three different attributes have been generated to define player behaviours: the distributions of the tower position in the *X* and *Y* axes, and the distribution of euclidean distances from the towers to the entry point.

- *Zigzag* distribution, Figure 3.2. This strategy has the characteristic that is developed along the *X* dimension, in the *Y* dimension the towers are placed in a fixed rank of positions. This way of placing towers makes the three distributions are uniform.
- *Vertical* distribution, Figure 3.3. This strategy has the towers place along the *Y* axis, this distributions has Gaussian distribution. Furthermore, the distribution of the tower position in the *X* axe has grouped the majority of the information in one bin, this happens because the *X* coordinates of the towers are all the same.
- *Horizontal* distribution, Figure 3.4. This strategy has the characteristic that is developed along the *X* dimension, it can be observed that the distribution of the dimension *X* and the distribution of the euclidean distance is uniform and the information from the dimension *Y* can be grouped in one bin.

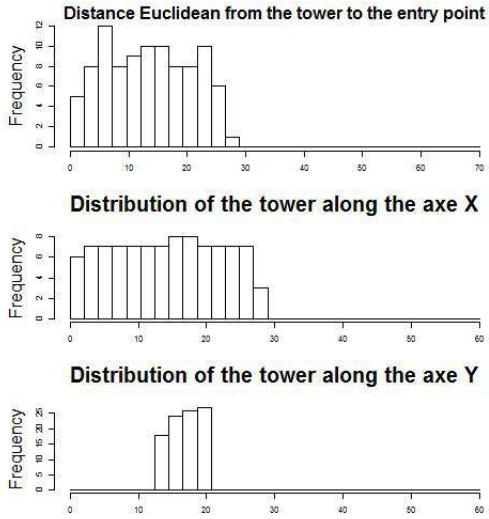

Fig. 3.2 Zigzag distribution

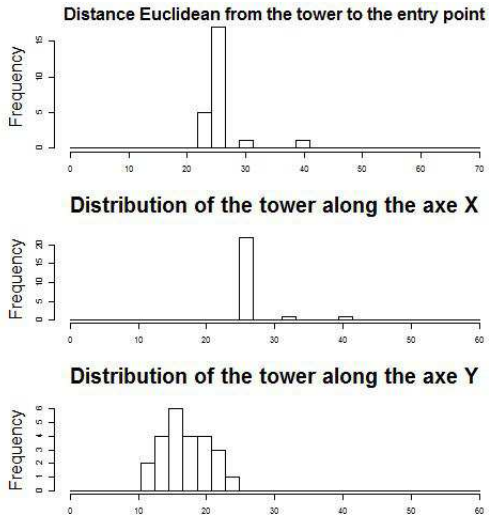

Fig. 3.3 Vertical distribution

3.4.2 The Shortest Path Between the Starting Point and the Exit Point

The video game has a board where the towers are placed. This board has two important points: the starting point and exit point. The starting point is where enemies enter in the board and the aim of the enemies is to reach the exit point. The player puts towers on the board to avoid that the enemies reach the exit point. Then, the enemies must find the shortest path between the starting point and the exit point to receive less damage. This shortest path changes in term of the strategy applied. To calculate the shortest path we use the board positions to create a graph. With the graph the Dijkstra algorithm is used to obtain the minimum path.

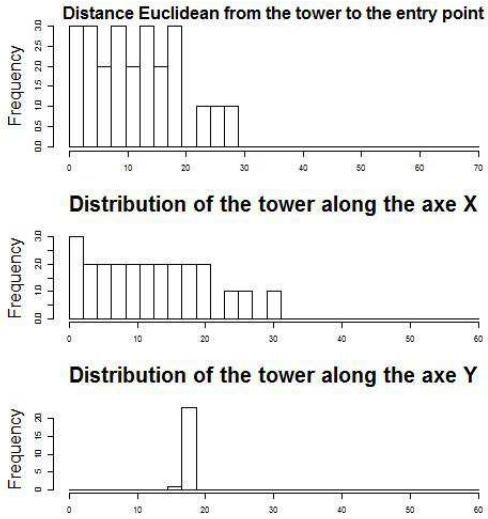

Fig. 3.4 Horizontal distribution

In the figure 3.5, can be see observe that the enemies in Zigzag strategy walk a larger path than in the Vertical or *Horizontal* Strategy.

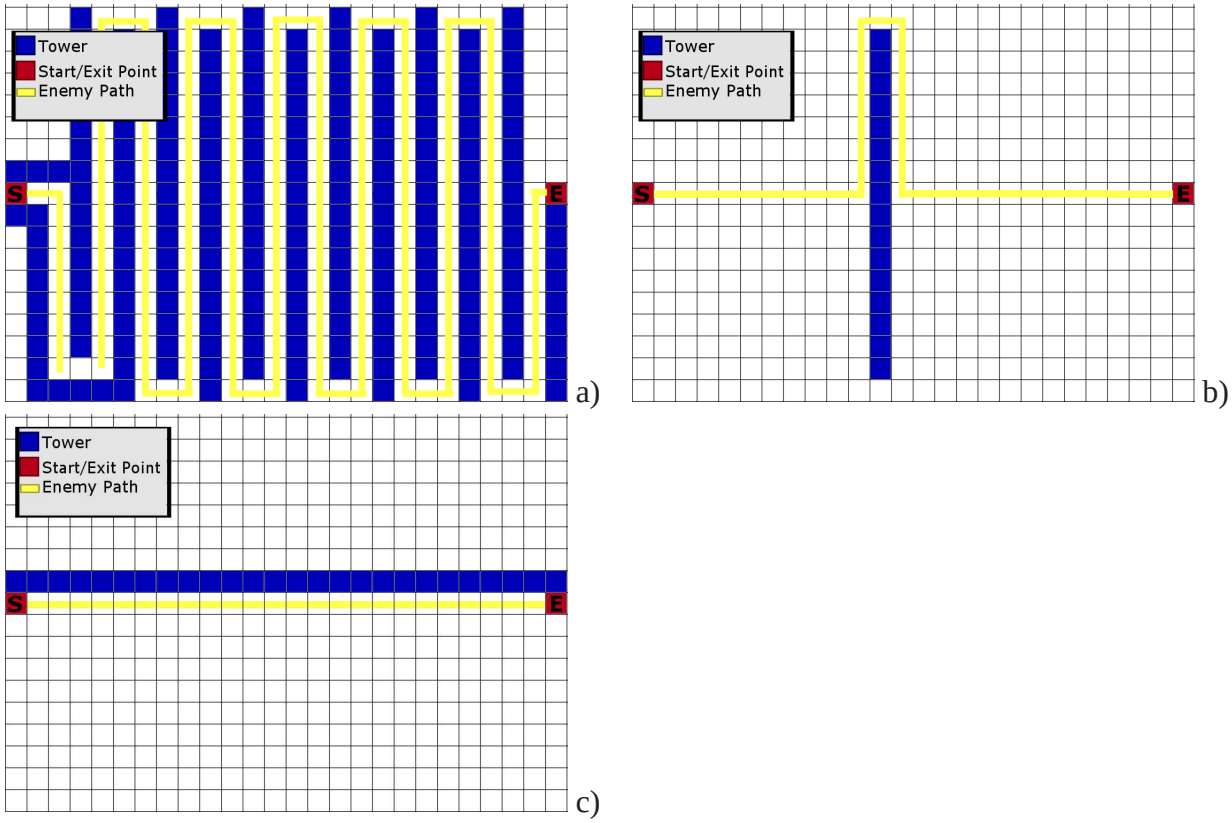

Fig. 3.5 Figure *a, b and c* show the shortest path in a Zigzag, Vertical and Horizontal strategy

3.4.3 Number of Turns

The section 3.4.2 shows how the strategies modify the path followed by enemies, it is different for each strategy, then the number of turns in the way depends on the strategy selected. The number of turns in a path are the amount of times an enemy changes the direction to reach the exit point. In the case of the *Zigzag* (see figure 3.6 *a*) strategy the number of turns is bigger than *Vertical* (see figure 3.6 *a*) strategy and for Horizontal the number of turns are 0.

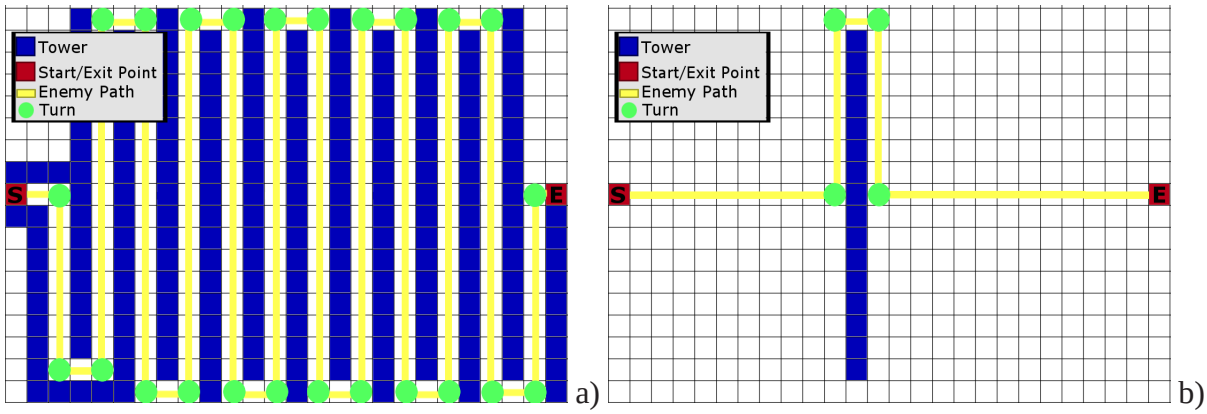

Fig. 3.6 Figure *a* and *b* show the turns that an enemy does in a Zigzag and Vertical.

3.4.4 Kill Ratio

Kill Ratio attribute measures the number of enemies destroyed per second by the user. First, we have generated time series using this attribute. Once we obtain the time series we calculate the autocorrelation. The autocorrelation of the time series provides a pattern which is used to distinguish the different strategies. In the figure 3.7, the sub-figures show in the first picture the autocorrelation pattern and the second picture the tower distribution of the strategy.

3.5 Data Analysis Procedure

Three main techniques have been used to obtain the players strategies. The first one is based on an sliding-window technique that is used to gather data and create instances of features. The second one is based on K-means clustering to group distributions by labels. The last technique studies the similitude among the distributions. This section describes these methods.

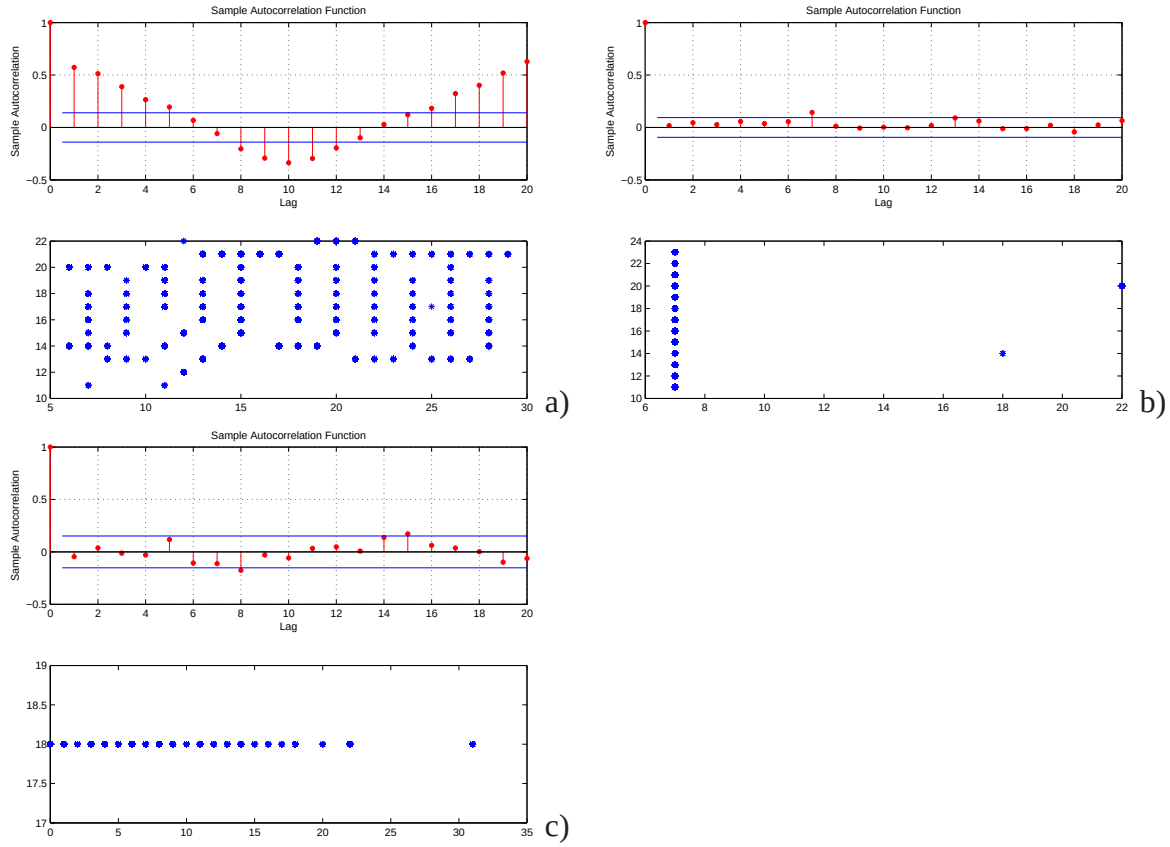

Fig. 3.7 Figure *a, b and c* show the kill/ratio in a Zigzag, Vertical and Horizontal strategy

3.5.1 Sliding-Window Technique

The most popular approach to deal with data stream involves the use of sliding windows algorithm [118]. This algorithm provides a way to divide the data stream in an amount of examples to analyse. The procedure of using sliding windows for data stream mining is shown in Algorithm 1. The input of the algorithm are the samples from the Tower Defence Game. One sample corresponds to one window, the size of the window is dynamic and it changes according to the life time of the wave. The life of the wave is defined as the time between the apparition of the first horde and the disappearance of the last horde. With the size of the windows defined, in each iteration of the algorithm, a new window is analysed and the distribution of the X coordinate , the distribution of the Y coordinate and the distribution of the euclidean distance from the towers to the exit are returned.

Algorithm 1: This algorithm is an adaptation of [71]

Parameter: $\mathcal{S}$: a data stream of example $\mathcal{W}$: window of examples
Result: C: the distribution of the coordinate X, the distribution of the coordinate Y and the distribution of the euclidean distance from the towers to the exit from the window $\mathcal{W}$

```
Initialize window W
forall the example x_i ∈ S do
    W ← W ∪ {x_i}
    build C using W
end
```

3.5.2 K-Means Clustering

K-means algorithm is used to partition the input data set into k partitions. However, k-means algorithm has two problems. The first one, in contrast to other algorithms, k-means cannot be used with arbitrary distance functions or be use on non-numerical data. And the second, K-Means algorithm can not guarantee finding the best space partition.

To solve the first problem we use the euclidean distance and transforms all dataset to numerical data. For the second, we execute the algorithm using always the same *'k'* several times (see Algorithm 2) and then we choose the best result returned. For this selection, we use two metrics: intra-cluster distance and inter-cluster distance. Intra-cluster [105] distance measure(equation 3.2) the average of the distances between the points and its respective cluster centroids. In the equation 3.2 we can see the intra-cluster metric, N is the number of instances of data extracted from the game, K is the number of clusters, and z_i is the centroid of cluster C_i.

$$intra(x, z_i) = \frac{1}{N} \sum_{i=1}^{K} \sum_{x \in C_i} ||x - z_i||^2 \tag{3.2}$$

$$inter(z_i, z_j) = min||z_i - z_j||^2; i = 1, 2, ..., K-1; j = i+1, ..., K; \tag{3.3}$$

Inter-cluster distance (equation 3.3) or the distance between clusters measures the distance between cluster centres. To choose the result that do a good partition of the data space, it is necessary to minimize the intra-cluster distance and maximize the inter-cluster distance measure *see* Fgure 3.8). The aim is to minimize the validity measure (equation 3.4).

$$validity(x, z_i, z_j) = \frac{intra}{inter} \tag{3.4}$$

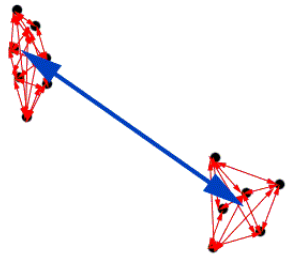

Fig. 3.8 This figure shows an example of 2 cluster where the big row (the blue one) represents the intra-cluster distance and the little rows (the red ones) are the inter-cluster distance

Algorithm 2: Algorithm to choose the best K-Means partitioning

Parameter: $\mathcal{W}$: window of examples
Result: C: data labelled in the window $\mathcal{W}$

```
1 Initialize window W
2 k=4
3 vecValidity ← []
4 for j = 1 to 10 do
5 |   labels ← KMeans(k, W)
6 |   validity ← CalculateValidity(labels, W)
7 |   vecValidity(i) ← validity
8 end
9 vecK(k) ← min(vecValidity)
```

3.5.3 Spectral Clustering

The goal of spectral clustering is to cluster data that is connected. Spectral clustering refers to a class of techniques, which rely on the eigenstructure of a similarity matrix to partition points into disjoint clusters. The points in the same cluster have high similarity and points in different clusters have low similarity.

Spectral clustering has many applications [94–96] in machine learning, exploratory data analysis, computer vision and speech processing. Most techniques explicitly or implicitly assume a metric or a similarity structure over the space of configurations, which is then used by clustering algorithms. The success of such algorithms depends heavily on the choice of the metric, but this choice is generally not treated as part of the learning problem.

To this work the programming language R has been used, because has a package named *kernlab* which it is define the Spectral Clustering method. The name of the function is *specc*. The arguments of functions are:

- *x*: the matrix of data to be clustered, or a symbolic description of the model to be fit, or a kernel Matrix of class kernelMatrix, or a list of character vectors.

- *data*: an optional data frame containing the variables in the model. By default the variables are taken from the environment which *specc* is called from.
- *centers*: Either the number of clusters or a set of initial cluster centers. If the first, a random set of rows in the eigenvectors matrix are chosen as the initial centers.
- *kernel*: the kernel function used in computing the affinity matrix. This parameter can be set to any function, of class kernel, which computes a dot product between two vector arguments.
- *kpar*: a character string or the list of hyper-parameters (kernel parameters). The default character string "automatic" uses a heuristic to determine a suitable value for the width parameter of the RBF kernel. The second option "local" (local scaling) uses a more advanced heuristic and sets a width parameter for every point in the data set. This is particularly useful when the data incorporates multiple scales.

3.5.4 Similitude

The different $\mathcal{W}$ that compose the strategies are labelled by groups with K-Means to study the similitude between distributions. Equation 3.5 represents the similitude between two distributions D_1 and D_2. In this equation, w_i represents the wave number, and $D_1(w_i)$ and $D_2(w_i)$ indicate the group of the distribution in w_i. The similitude is calculated dividing the number of the label coincidences from the distributions ($D_1(w_i)$ and $D_2(w_i)$) between the number of waves.

$$Similitude(D_1, D_2) = \frac{\#\{i \epsilon \{1 \ldots \#Waves\} | D_1(w_i) = D_2(w_i)\}}{\#Waves} \tag{3.5}$$

Chapter 4

Experimentation

The experimental phase is divided into three phases. In the first phase, we discuss the different types of strategies that have been used by the players to classify different behaviours. In the second phase, we study the similitude between the strategies detected in the previous phase, in order to get an instrument to determine empirically whether the distributions are good discriminants. In the last phase, we use the representative behaviour operators of each distribution (the centroids obtained in the previous phase by K-Means) as input to generate new adaptive hordes inside a gameplay, and then measure the user satisfaction. This section describes these phases.

4.1 Experiment #1: Visual Strategy Identification

In this initial approach, only the towers on the game board are analysed when different types of wave are generated. The X axis represents the width and the Y represents the height (the size of X is 66 and the size Y is 33). The starting point where the enemies are generated is the position $(0,17)$ and the exit point is located in $(66,17)$. In each different wave of a gameplay, only one horde of enemies is generated. The enemies of these hordes have the same stamina and strength, but the number of enemies could change during the game. Changing the size of the hordes, two different experiments have been carried out (constant and linear) to determine the user strategies.

The gameplay is divided in 10 waves and each one has two different phases. The first phase has no enemies and the gamer has 5 seconds to put or remove towers in the board. During the second phase the enemies will try to reach the exit, so the user will need to put more towers, remove the existing ones in the board. In this second phase the enemies will appear with a frequency of 0.5 seconds, and the number of enemies will be generated

according to equation 3.1. Once both phases have concluded there are only two possibilities, a successful strategy (if the player is able to destroy all the enemies), or a failed strategy (if at least three o more enemies are able to reach the exit).

The dataset used is composed by 24 gameplays, 12 of them with a constant horde size in each wave and the other 12 with linear growth of the horde size. In the experiment the gamers, which play for first time the Tower Defence Game, have been divided in two groups: in the first one, the players have been assigned strategies, and in the second one, gamers play without an assigned strategy. Each gamer plays three times in the first experiment and three times in the second one. Once the first group strategies have been analysed, the strategies of the second group are identified.

Using the visualization tool, the histogram, with the data extracted from the different gameplays we have found four strategies in the first group,:

- The first one, *Zigzag* distribution, is useful to slow the enemies. The goal is that enemies spend longer time to reach the exit point, therefore the towers have more time to shot them, see figure 4.1. This strategy has the characteristic that is developed along the X dimension, while in the Y dimension the towers are placed in a fixed rank of positions. This way of placing towers makes the three distributions uniform.

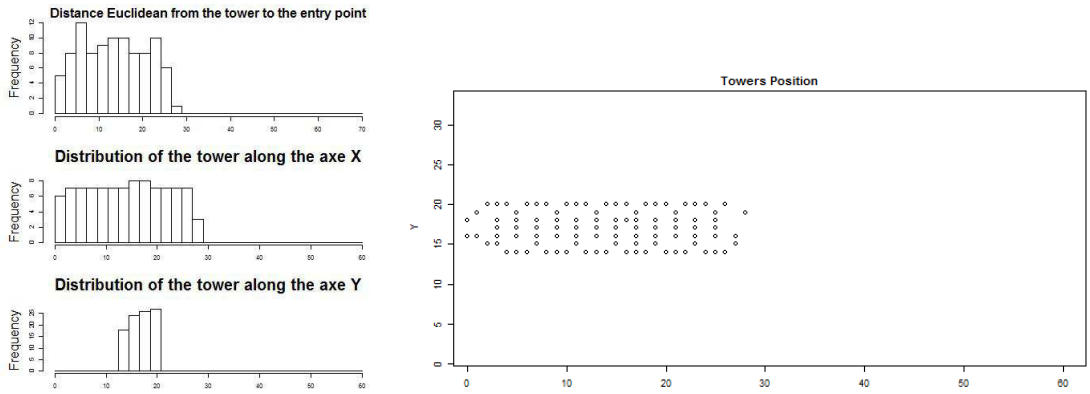

Fig. 4.1 Zigzag distribution

- The second one, *Vertical* distribution has the characteristic that towers are distributed along the Y dimension in one or two columns. This strategy concentrates the strength of the towers in a big column cluster to destroy enemies, see figure 4.2. In this strategy we can observe that towers are placed along the Y axis. This dimension has Gaussian distribution. Furthermore, the distribution on the X dimension has the majority of the information grouped in one bin. This happens because the coordinates X of the towers are all the same.

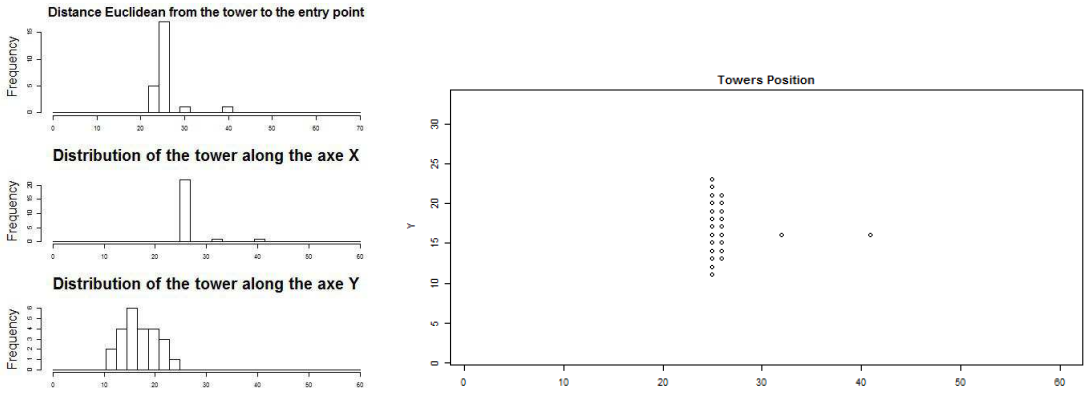

Fig. 4.2 Vertical distribution

- The third one, *Grouped* distribution, joins towers in small clusters to increase the damage that enemies receive. These clusters are spread throughout the game board, see figure 4.3. The clusters can have different aspects: square, circular, lengthened, etc. This strategy has the characteristic that histograms have a saw distribution.

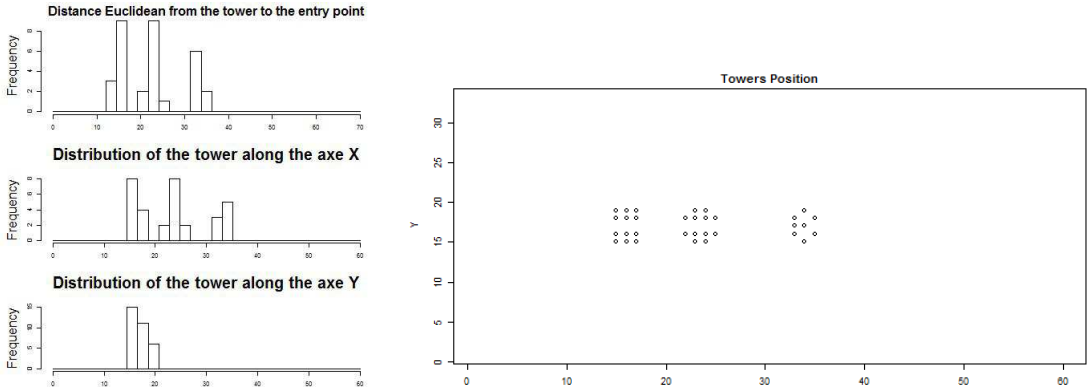

Fig. 4.3 Grouped distribution

- Finally, *Horizontal* distribution, creates a horizontal wall of towers across the game board. This strategy is similar to *Vertical* strategy with a orientation change. The aim is to concentrate the strength of the towers in a big horizontal cluster to destroy the enemies, see figure 4.4. This strategy has the characteristic that is developed along the X dimension. We can observe that the distribution of the X dimension and the distribution of the euclidean distance is uniform and the information from the Y dimension can be grouped in one bin.

Using the histogram to analyse the gameplays some conclusion can be given. *Horizontal* and *Vertical* distributions are easy to identify. When a *Horizontal* distribution is used, towers

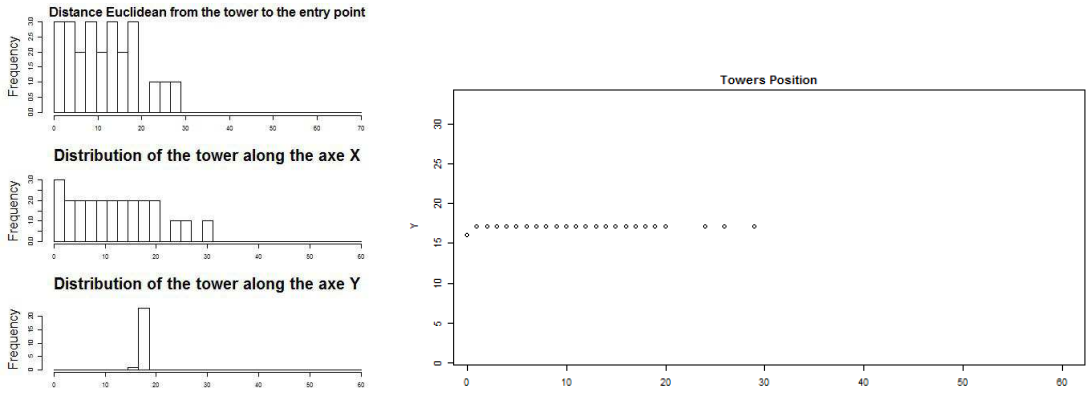

Fig. 4.4 Horizontal distribution

are concentrated in axis *Y* in one bin. However, in *Vertical* distributions, this effect appears in axis *X*. The same occurs with the *Grouped* distribution, which has a saw distribution in the axis *X* and *Y* that is easy to recognize. Finally, *Zigzag* distribution is difficult to identify because the three distributions are uniform and there are several ways to place towers that can create this kind of distribution. For example, in the *Horizontal* distribution, both the euclidean distance and the *X* axis, has an uniform distribution. It is easy to confuse both strategies. On the other hand, the figure 4.6 presents a game where the gamer does not use a *Zigzag* tactic but the distribution (see figure 4.5) is similar to *Zigzag*. Furthermore, analysing the *Grouped* distribution we observed that the rest of the strategies are a particular case of *Grouped* distribution. For example, if the player grouped the towers in groups that are near among them along the *X* axe, the strategy used is very similar to the *Horizontal* distribution (see Figure 4.7). The same happens if we put the cluster around the *Y* axe, the strategy used is similar to the *Vertical* distribution. For this reason, we reduce the number of strategies from 4 to 3 (*Horizontal*, *Vertical* and *Zigzag* distributions).

4.2 Experiment #2: Data Analysis using Clustering algorithm

To study the similitude among strategies, the first step consist in grouping the data of the gameplays in groups. All the data of one group have the same characteristics. In this case, the number of groups is 3 because there are 3 different strategies Clustering algorithms are adopted to separate the data in groups. The data of the groups have been previously analysed and labelled according to the strategies identified. With the output of the clustering techniques we calculate the similitude among strategies.

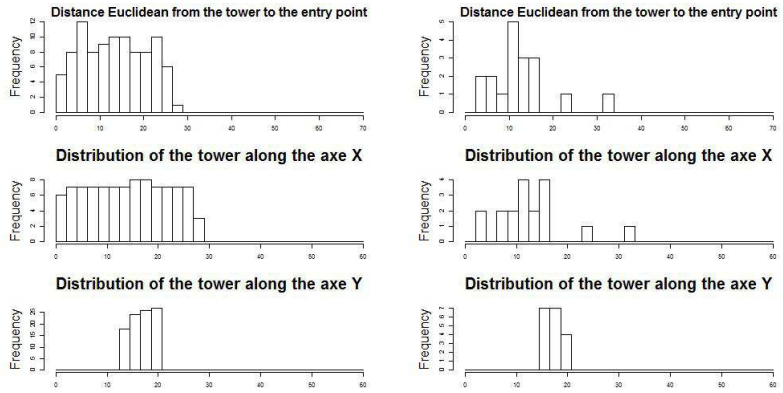

Fig. 4.5 Fig (*a*) shows the Zigzag distribution, whereas Fig (*b*) shows the distribution from one of the second group gamers

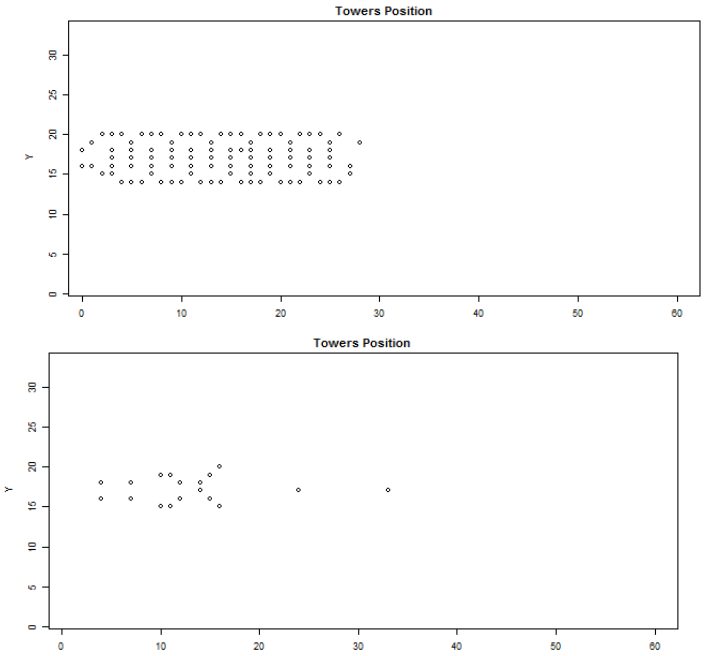

Fig. 4.6 Fig (*a*) shows how the player places towers using Zigzag distribution, whereas Fig (*b*) shows the towers position from the second group of gamers.

Sliding-window technique has been used to extract the ten wave distribution of a gameplay. The features are grouped by labels that are assigned by *K-Means* (see table 4.1) and Spectral Clustering (see table 4.2) algorithm, these labels denote that the different gameplay waves has characteristic in common. As it was aforementioned above in section 4.1, we identify 3 strategies, and for this reason we choose $k = 3$. If we study the tables, we can observe that $k = 3$ always is not the correct value. The wave 6 in table 4.1 there is only one

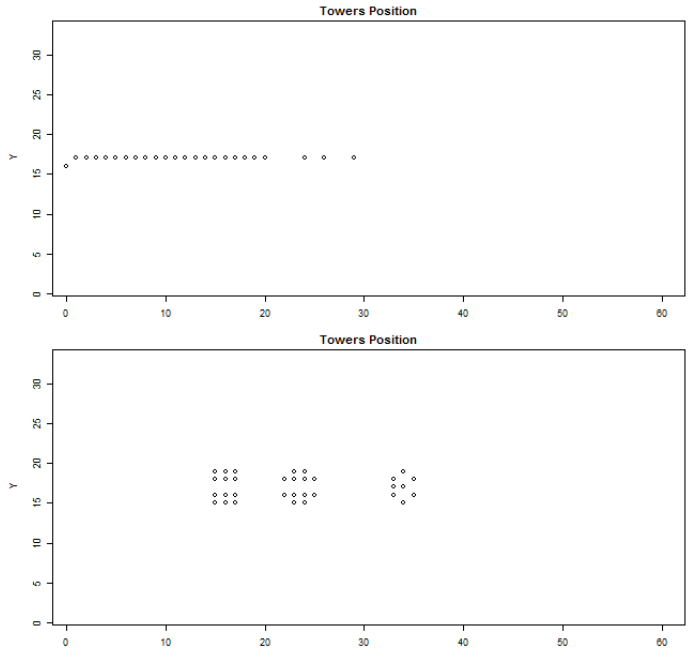

Fig. 4.7 In this figure, it is possible appreciate that the *Horizontal* distribution (subfigure a) is very similar to the Gruoped distribution (subfigure (b))

label of type 1, the same occurs in the wave 10 in table 4.2. This means, the number of cluster in these type of waves are lower than 3. This happens because there are strategies in some waves that have similar patterns and the clustering algorithms group together. As a future work is necessary to study more attributes and the best *k* in each wave.

4.3 Analysing of the Similitude among K-Means Gameplays

In this section, the similitude among strategies have been studied using the tables 4.1 and 4.2 and the equation 3.5, the results appear in the tables 4.6 and 4.7.

In the tables 4.6 and 4.7, it can be observed that similitude between different gampeplays using the same strategy is high. This means that the attributes correctly define the strategies. If we compare both tables we observe that Spectral Clustering algorithm has the best similitude between gameplays that use the same strategy.

Taking into account at the similitude between different strategies, *K-Means* distinguish between *Zigzag* and *Horizontal* strategies, and between *Zigzag* and *Vertical* strategies. But

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	H	H	H	H	V	V	V	V	V	Z	Z	Z	Z	Z
Wave 1	2	2	2	2	2	2	2	2	2	2	3	3	1	1	1
Wave 2	3	3	3	3	3	3	3	3	1	1	1	1	2	1	1
Wave 3	2	2	2	2	2	2	3	3	3	3	1	1	1	1	1
Wave 4	1	1	3	3	3	3	1	1	1	1	1	2	2	2	2
Wave 5	3	3	3	3	3	3	3	3	3	3	2	2	1	1	1
Wave 6	3	3	3	3	3	3	3	3	1	3	2	2	3	2	3
Wave 7	1	1	1	1	1	1	1	1	2	2	2	3	3	1	1
Wave 8	2	2	2	3	3	3	3	1	1	1	1	1	1	1	1
Wave 9	1	3	1	1	1	1	1	1	3	3	3	2	2	2	2
Wave 10	3	1	1	1	1	2	2	2	2	2	2	2	2	2	2

Table 4.1 Representation of the distributions labelled by *K-Means*, where Z is the Zigzag distribution, H is the Horizontal distribution and V is the Vertical distribution.

the similitude between *Horizontal* and *Vertical* strategies is high. This occurs because *K-Means* algorithm does not take into account the geometric shape of the data. For *K-Means*, *Horizontal* and *Vertical* strategies have the same shape.

The table 4.7 shows that *Spectral Clustering* differentiate between *Vertical* and *Horizontal* strategies and between *Horizontal* and *Zigzag* strategies. But the similitude between *Zigzag* and *Vertical* strategies is high. This occurs because *Spectral Clustering* algorithm takes into account the geometric shape of the data. In *Zigzag* strategy the towers are positioned along the Y axis, as walls, to create a crossroad and *Vertical* put only one wall along the Y axis. For this reason *Spectral Clustering* identifies *Vertical* strategies as a particular case of *Zigzag*.

In conclusion, both algorithms distinguish correctly between gameplays with the same strategy but *Spectral Clustering*, in this case, does the best partition. On the other hand, *K-Means* differentiate better than *Spectral Clustering* gameplays with different strategies. The table 4.6 shows that the similitude between *Horizontal* and *Vertical* strategies is high in *K-Means*, but the similitude between *Zigzag* and *Vertical* strategies in the table 4.7 shows that is highest to *Spectral Clustering*. So, we can say that *K-Means* do a better data partition than *Spectral Clustering*.

4.4 Precision of the Clustering Results

Once the table of similitude is computed (see section 4.3) it is necessary a metric that helps to easily identify which attributes combination is the best to define the player strategies.

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	H	H	H	H	V	V	V	V	V	Z	Z	Z	Z	Z
Wave 1	1	1	1	1	1	2	2	2	2	2	3	3	1	2	1
Wave 2	2	3	3	3	3	1	1	1	1	1	1	1	1	1	1
Wave 3	1	1	2	1	3	3	3	3	3	3	3	3	3	3	3
Wave 4	2	2	1	2	2	3	3	3	3	3	3	3	3	3	3
Wave 5	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3
Wave 6	1	1	1	1	2	3	3	3	3	3	3	3	3	3	3
Wave 7	2	3	2	2	2	1	1	1	1	1	1	1	1	1	1
Wave 8	1	1	1	1	2	3	3	3	3	3	3	3	3	3	3
Wave 9	1	2	1	1	1	3	3	3	3	3	3	3	3	3	3
Wave 10	2	3	3	1	3	2	2	2	2	2	2	2	2	2	2

Table 4.2 Representation of the distributions labelled by Spectral Clustering, where Z is the Zigzag distribution, H is the Horizontal distribution and V is the Vertical distribution.

This metric is the precision metric(see equation 4.3). The parameters of the metric are: the True Positive (*TP*, see equation 4.1) value, the False Positive (*FP*, see equation 4.2) value and the number of the player strategies (*K*).

$$TP = \sum_{\substack{0<i<k \\ 0<j<k}} \frac{S_{i,j}}{K} \tag{4.1}$$

$$FP = \sum_{\substack{0<i<k \\ 0<j<k \\ i<j}} \frac{S_{i,j}}{K} \tag{4.2}$$

$$Precision = \frac{TP - FP}{K} \tag{4.3}$$

The figure 4.9 shows an example where the precision metric is applied. The similitude table is used to calculate the values of the precision parameters. This matrix is divided in submatrix (S) of size 5×5 (see figure 4.8), 5 is the number of gameplays of each strategy. The submatrix 1.1, 2.2 and 3.3 have the similitude values between gameplays with the same strategy and the submatrix 1.2, 1.2 and 2.3 have the similitude values between gameplays with different strategy. Once the matrix is divided in the different submatrix, it is possible calculate the TP and FP values. TP is equal to the sum of the average from the diagonal elements, and FP is equal to the sum of the average from upper triangular elements.

In general Spectral Clustering offers better results than K-Means. The table 4.5 shows

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	H	H	H	H	V	V	V	V	V	Z	Z	Z	Z	Z
H	1.0	0.8	0.8	0.7	0.7	0.7	0.7	0.7	0.3	0.4	0.1	0.0	0.1	0.1	0.2
H	0.8	1.0	0.8	0.7	0.7	0.6	0.6	0.6	0.4	0.5	0.2	0.0	0.1	0.1	0.2
H	0.8	0.8	1.0	0.9	0.9	0.8	0.6	0.6	0.2	0.3	0.0	0.0	0.1	0.1	0.2
H	0.7	0.7	0.9	1.0	1.0	0.9	0.7	0.6	0.2	0.3	0.0	0.0	0.1	0.1	0.2
H	0.7	0.7	0.9	1.0	1.0	0.9	0.7	0.6	0.2	0.3	0.0	0.0	0.1	0.1	0.2
V	0.7	0.6	0.8	0.9	0.9	1.0	0.8	0.7	0.3	0.4	0.1	0.1	0.2	0.2	0.3
V	0.7	0.6	0.6	0.7	0.7	0.8	1.0	0.9	0.5	0.6	0.2	0.1	0.2	0.2	0.3
V	0.7	0.6	0.6	0.6	0.6	0.7	0.9	1.0	0.6	0.7	0.3	0.2	0.3	0.3	0.4
V	0.3	0.4	0.2	0.2	0.2	0.3	0.5	0.6	1.0	0.9	0.6	0.3	0.2	0.3	0.3
V	0.4	0.5	0.3	0.3	0.3	0.4	0.6	0.7	0.9	1.0	0.6	0.3	0.3	0.3	0.4
Z	0.1	0.2	0.0	0.0	0.0	0.1	0.2	0.3	0.6	0.6	1.0	0.7	0.3	0.5	0.4
Z	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.3	0.7	1.0	0.6	0.7	0.6
Z	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.2	0.3	0.3	0.6	1.0	0.7	0.8
Z	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.5	0.7	0.7	1.0	0.9
Z	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.4	0.3	0.4	0.4	0.6	0.8	0.9	1.0

Table 4.3 Representation of the *K-Means* similitude between distributions, where Z is the Zigzag distribution, H is the Horizontal distribution and V is the Vertical distribution.

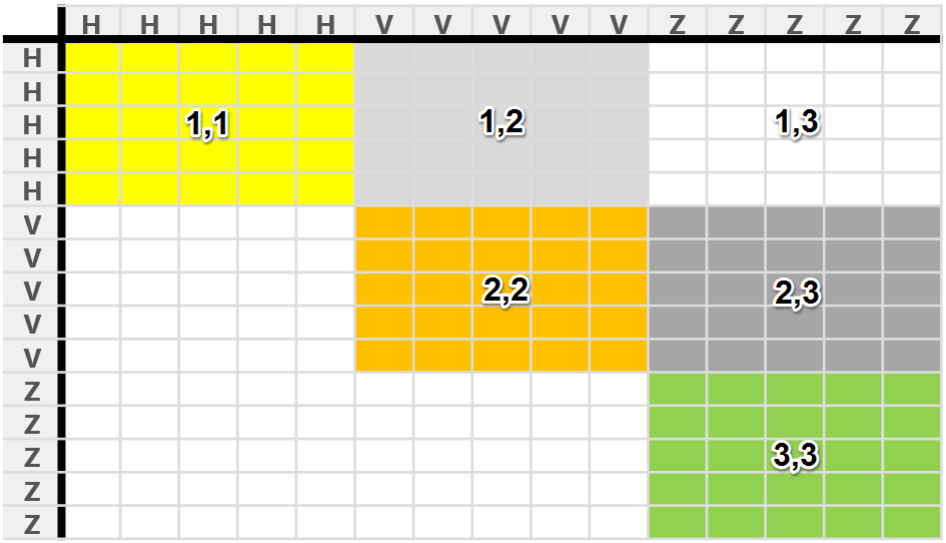

Fig. 4.8 Example of submatrix

that Spectral Clustering has 5 combinations of attributes with a value of *precision* greater than 0.6. While in K-Means the max value of precision is 0.52.

Although Spectral Clustering has better result than K-Means, there are some case where the opposite happens. An example are the attribute *Kill*/*Ratio* and the attribute combina-

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	H	H	H	H	H	V	V	V	V	V	Z	Z	Z	Z	Z
H	1.0	0.6	0.6	0.8	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.2
H	0.6	1.0	0.6	0.7	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1
H	0.6	0.6	1.0	0.7	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1
H	0.8	0.7	0.7	1.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1
H	0.5	0.5	0.6	0.6	1.0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.2
V	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
V	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
V	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
V	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
V	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
Z	0.1	0.0	0.0	0.0	0.1	0.9	0.9	0.9	0.9	0.9	1.0	1.0	0.9	0.9	0.9
Z	0.1	0.0	0.0	0.0	0.1	0.9	0.9	0.9	0.9	0.9	1.0	1.0	0.9	0.9	0.9
Z	0.2	0.1	0.1	0.1	0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	1.0
Z	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9	1.0	0.9
Z	0.2	0.1	0.1	0.1	0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	1.0

Table 4.4 Representation of the Spectral Clustering similitude between distributions, where Z is the Zigzag distribution, H is the Horizontal distribution and V is the Vertical distribution.

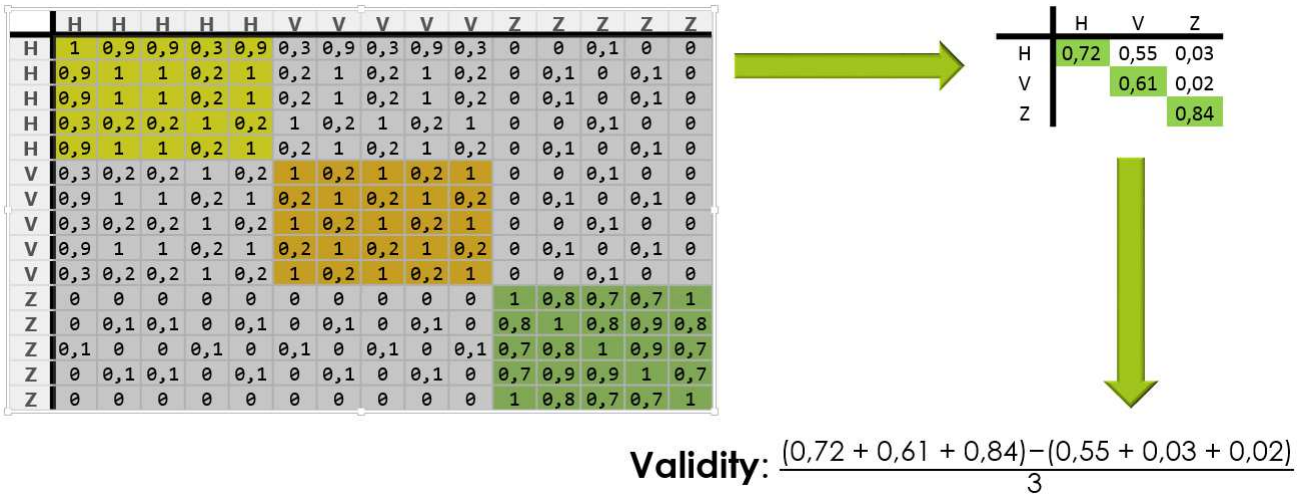

Fig. 4.9 Example of precision metric

tion of the $X + Y + DistEU + Kill/Ratio$. As can be appreciate in both cases the attribute $Kill/Ratio$ appears. This means that the attribute $Kill/Ratio$ add some noise for the Spectral Clustering method.

Now, analysing only the K-Means results (see table 4.6) we can observe that the attribute

		Precision
Atributes	K-Means	Spectral
X, Y, DistEU, PathDist, Kill/Ratio	0,06	**0,67**
X, Y, DistEU, PathDist, Turns	0,02	**0,65**
X, Y, DistEU, Kill/Ratio	**0,52**	0,10
X, Y, DistEU, PathDist	0,04	0,59
X, Y, DistEU	**0,46**	**0,73**
X, Y, PathDist	0,04	**0,71**
X, Y	**0,50**	**0,71**
PathDist, Turns, Kill/Ratio	0,06	0,52
PathDist, Turns, DistEU	0,02	0,09
PathDist, Turns	0,06	0,10
PathDist, DistEU	0,04	0,36
Kill/Ratio	0,20	0,16

Table 4.5 **X:** X distribution. **Y:** Y distribution. **DistEU:** The distribution of the euclidean distance between the towers and the exit point. **PathDist:** The shortest path between the starting point and the exit point. **Kill/Ratio:** The number of enemies destroyed per time-step. **Turns:** The times an enemy must change the direction

	Precision
Atributes	K-Means
X, Y, DistEU, Kill/Ratio	0,52
X, Y	0,50
X, Y, DistEU	0,46
kill/Ratio	0,20
X, Y, DistEU, PathDist, Kill/Ratio	0,06
PathDist, Turns, Kill/Ratio	0,06
PathDist, Turns	0,06
X, Y, DistEU, PathDist	0,04
DistEU, PathDist	0,04
X, Y, PathDist	0,04
X, Y, DistEU, PathDist, Turns	0,02
DistEU, PathDist, Turns	0,02

Table 4.6 K-Means validation results

PathDist combined with other attributes add noise, as a consequence the *precision* value decrease. With the *DistUE* occurs the same, if we combine this attribute with *X* and *Y* attributes the precision value decrease from 0.5 to 0.46. Instead, *Kill*/*Ratio* attribute has a value of 0.20, this implies that the attribute give enough information about the strategy

type. Also, the attribute *X* and *Y* have a precision value of 0.50. When *Kill*/*Ratio* is joined with the *X*, *Y* and *DistEU* attributes the precision value is 0.52. The value, 0.52, is better than precision value of *X* and *Y*, 0.5. This happens because Kill/Ratio gives additional information which helps to identify the players strategies. After the table 4.6 analysis we conclude that a possible good combination of attributes is *X*, *Y* and *Kill*/*Ratio*.

	Precision
Atributes	Spectral
X, Y, DistEU	0,73
X, Y, PathDist	0,71
X, Y	0,71
X, Y, DistEU, PathDist, Kill/Ratio	0,67
X, Y, DistEU, PathDist, Turns	0,65
X, Y, DistEU, PathDist	0,59
PathDist, Turns, Kill/Ratio	0,52
DistEU, PathDist	0,36
kill/Ratio	0,16
PathDist, Turns	0,10
X, Y, DistEU, Kill/Ratio	0,10
DistEU, PathDist, Turns	0,09

Table 4.7 Spectral validation results

Finally, we study the result of the Spectral Clustering from the table 4.7. As it was aforementioned before, this method offers better results than K-Means. The table 4.7 shows that less information help to identify player strategies are *Kill*/*Ratio* and the combination of *PathDist*+*Turns* and *PathDist*+*DistEU*+*Turns*. With the values 0.16, 0.10 and 0.09 respectively. When we combine *PathDist*+*Turns* or *PathDist*+*DistEU*+*Turns* with *X*+*Y*+*DistEU*, the precision value is lower than *X*+*Y*+*DistEU* value. This happens because the attributes add noise.

On the other hand, in the table 4.7 there are values greater than 0.6. The best combination are *X*+*Y* and *X*+*Y*+*DistEU*. We do not consider the combination of *X*+*Y*+*PathDist* because has the same value of *X*+*Y* combination, this means that the attribute *PathDist* do not provide additional information. As in K-Means method the attributes with better performance are *X* and *Y*.

In conclusion, Spectral have better performance than K-Means in general. In both methods we can see that the attribute *PathDist* add noise and it is not good to identify strategies. While, the attributes *X* and *Y* give a lot of information about the player strategies. Further-

more, there are attributes that in one method put noise and in other add information about the player strategy, i.e. *DistEU* attribute.

Chapter 5

Conclusions & Future Work

This work provides an initial study on gamers interaction and their strategies used in a Tower Defence game. To achieve this purpose, a framework based on an open-source Tower Defence platform has been designed four us. Two experiments have been carried out to study the players interaction: in the first experiment, we use the visualization techniques to identify the different strategies. And in the second one, we study the best combination of attributes that define correctly the player strategies.

In the first experiment, four different strategies have been detected: *Zigzag* strategy, *Vertical* strategy, *Grouped* strategy and *Horizontal* Strategy. Using visualization techniques (histogram) we see ,on one hand, that the towers positions are not enough to identify the players strategies and ,on the ohter hand, that the Zigzag, Horizontal and Vertical distributions are a particular case of Grouped distribution. For this reason the number of strategies has been reduced to 3: Zigzag, Horizontal and Vertical distributions

The second experiment determines if the attributes selected to model the players behaviour are sufficiently descriptive. For this purpose, we use K-means and Spectral Clustering algorithms to group the strategies. Latter these groups have been used to compute the similitude among gameplays and using an evaluation function were determined with is the best combination of attributes that best fit the player strategies. From the similitude study, we have concluded that the attributes that best define the strategies are the towers positions.

Finally, the last experiment studies the impact on players entertainment (satisfaction). Although the developed adaptive horde generation makes the OTD more amusing, due to the bad strategy detection, the user satisfaction is not always achieved, as some gameplays are pretty easy and other are too difficult.

In future works, it will be necessary to use more features to perform a better classification and study more unsupervised techniques to determine which provide the best data

partitioning. Moreover, we could apply Online Learning, so the model is updated every time new data is gathered. On other hand, it is necessary to store the user gameplay records to customize the OTD in terms of users behaviour instead of the gameplay strategy.

Chapter 6

Contributions

During the development of these work the following contributions have been generated.

- F. Palero, A. Gonzalez-Pardo, and D. Camacho. "Simple gamer interaction analysis through tower defence games". International Conference on Computational Collective Intelligence Technologies and Applications 24th-26th September 2014, Seoul, Korea,(ICCCI 2014). In Proceedings New Trends in Computational Collective Intelligence (pp. 185-194). Springer International Publishing, Vol. 8733.

- F. Palero, C. Ramirez-Atencia, and D. Camacho. "Online Gamers Classification using K-means". International Symposium on Intelligent Distributed Computing, September 3-5, 2014, Madrid, Spain, (IDC 2014). In Springer International Publishing. In Proceedings Intelligence Distributed Computing VII (pp. 201-208). Springer International Publishing, Vol. 570.

- F. Palero, A. Gonzalez-Pardo, and D. Camacho. "Evaluación de Modelos de Jugadores mediante Técnicas de Clustering". Congreso de la Sociedad Española para las Ciencias del Videojuego Barcelona, 24 de junio de 2014, (COSECIVI 2014). (pp. 58-65). In Proceedings 1st Congreso de la Sociedad Española para las Ciencias del Videojuego.

References

[1] Ms pac-man software kit. Available in: http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html

[2] Platformer ai: Lever generator track. Available in: http://www.marioai.org/LevelGeneration

[3] Platformer ai: Turing test track. Available in: http://www.marioai.org/turing-test-track

[4] Ptsp framework. Available in: http://www.marioai.org/turing-test-track

[5] Skynet for protoss. Available in: http://code.google.com/p/skynetbot/

[6] Starcraft adjutant for terran. Available in: http://code.google.com/p/adjutantbot/

[7] Starcraft: Aiur for protoss. Available in: http://code.google.com/p/aiurproject/

[8] Starcraft: Bthai for terran. Available in: http://code.google.com/p/bthai/

[9] Starcraft: Bwapi for all razes. Available in: https://code.google.com/p/bwapi/

[10] Starcraft: Nova for terran. Available in: http://nova.wolfwork.com

[11] A. Abraham and V. Ramos. Web usage mining using artificial ant colony clustering and linear genetic programming. In Evolutionary Computation, 2003. CEC 03. The 2003 Congress on, volume 2, pages 1384 – 1391 Vol.2, dec. 2003.

[12] Adnan Acan. Gaaco: A ga + aco hybrid for faster and better search capability. In Ant Algorithms. Springer Berlin / Heidelberg, 2002.

[13] Lotte Berghman, Dries Goossens, and Roel Leus. Solving mastermind using genetic algorithms. Computers & Operations Research, 36:1880 – 1885, 2009.

[14] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. Evolutionary Computation, 4, no. 4:361–394, 1996.

[15] C. Blum and D. Merkle. Swarm Intelligence: Introduction and Applications. Springer Publishing Company, Incorporated, 1 edition, 2008.

[16] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford, 1999.

[17] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1, No. 1:1–23, 1993.

[18] Xingguo Chen, Hao Wang, Weiwei Wang, Yinghuan Shi, and Yang Gao. Apply ant colony optimization to tetris. Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO), 1:1741–1742, 2009.

[19] JI Chunlin. A revised particle swarm optimization approach for multi-objective and multi-constraint optimization. In GECCO, 2004.

[20] Jack Coldridge and Martyn Amos. Genetic algorithms and the art of zen. Technical report, Manchester Metropolitan University, 2010.

[21] Nicholas Cole, Sushi1 J. Louis, and Chris Miles. Using a genetic algorithm to tune first-person shooter bots. Proceedings of the Congress on Evolutionary Computation, 1:139 – 145, 2004.

[22] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In European Conference on Artificial Life, pages 134–142, 1991.

[23] G. Cormode. The hardness of the lemmings game, or oh no, more npcompleteness proofs. In In Proceedings of Third International Conference on Fun with Algorithms, pages 65–76, 2004.

[24] Mario J. Perez-Jimenez Agustin Riscos-Nuñez Daniel Diaz-Pernil, Miguel A. Gutierrez-Naranjo. A linear solution for subset sum problem with tissue p systems with cell division. Lecture Notes in Computer Science, 4527:170–179, 2007.

[25] S. Das, A. Biswas, S. Dasgupta, and A. Abraham. Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. Foundations of Computational Intelligence, 203:2355, 2009.

[26] Tridib Kumar Das. Bio-inspired algorithms for the design of multiple optimal power system stabilizers: Sppso and bfa. IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS, 44(5), September/October 2008.

[27] Erik D. Demaine, Susan Hohenberger, and David Liben-Nowell. Tetris is hard, even to approximate. In Proceedings of the 9th International Computing and Combinatorics Conference, 2003.

[28] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem, 1997.

[29] Marco Dorigo. Ant colony optimization: A new meta-heuristic. In Proceedings of the Congress on Evolutionary Computation, pages 1470–1477. IEEE Press, 1999.

[30] A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Springer-Verlag, 2009.

[31] A.P. Engelbrecht. Computational Intelligence: An Introduction. Wiley Publishing, 2nd edition, 2007.

[32] Ozgur B. Akan Falko Dressler. Bio-inspired networking: From theory to practice. IEEE Communications Magazine, pages 177–183, November 2010.

[33] M. Farooq. Bee-Inspired Protocol Engineering: From Nature to Networks. Springer Publishing Company, Incorporated, 2008.

[34] D. B. Fogel. Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, 1995.

[35] Stephanie Forrest. Genetic algorithms: principles of natural selection applied to computation. Science, 261, No. 5123:872–878, 1993.

[36] Tiaoping Fu, Yushu Liu, Jiguo Zeng, and Jianhua Chen. An improved genetic and ant colony optimization algorithm and its applications. In De-Shuang Huang, Kang Li, and George Irwin, editors, Intelligent Control and Automation, volume 344 of Lecture Notes in Control and Information Sciences, pages 229–239. Springer Berlin / Heidelberg, 2006. 10.1007/978-3-540-37256-1_31.

[37] Antonio Gonzalez-Pardo and David Camacho. A new csp graph-based representation for ant colony optimization. In 2013 IEEE Conference on Evolutionary Computation, volume 1, pages 689–696, June 20-23 2013.

[38] Antonio Gonzalez-Pardo and David Camacho. Environmental influence in bio-inspired game level solver algorithms. In 7th International Symposium on Intelligent Distributed Computing (IDC 2013)., volume 511 of Studies in Computational Intelligence, pages 157 – 162. Springer Berlin Heidelberg, 2014.

[39] Huang Guangdong, Ling Ping, and Wang Qun. A hybrid metaheuristic aco-ga with an application in sports competition scheduling. In Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007.

[40] Robin Houston, Joseph White, and Martyn Amos. Zen puzzle garden is np-complete. Information Processing Letters, 112:106–108, 2012.

[41] K. A. De Jong. Evolutionary computation-a unified approach. MIT Press, 2006.

[42] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Techn Rep TR06 Erciyes Univ Press Erciyes, 129(2):2865, 2005.

[43] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J. of Global Optimization, 39:459–471, 2007.

[44] Graham Kendall and Kristian Spoerer. Scripting the game of lemmings with a genetic algorithm. Proceedings of the Congress on Evolutionary Computation, 1:117 – 124, 2004.

[45] J. Kennedy and R. Eberhart. Particle swarm optimization. Proceedings of the Congress on Evolutionary Computation, 4:1942–1948, 1995.

[46] Pier Luca Lanzi, editor. Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milano, Italy, 7-10 September, 2009. IEEE, 2009.

[47] Zne-Jung Lee, Shun-Feng Su, Chen-Chia Chuang, and Kuan-Hung Liu. Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment. Applied Soft Computing, 8(1):55 – 78, 2008.

[48] Chong-U Lim, Robin Baumgarten, and Simon Colton. Evolving behaviour trees for the commercial game defcon. In EvoGames, 2010.

[49] Simon M. Lucas. Computational intelligence and ai in games: A new ieee transactions. T-CIAIG, 2011.

[50] Emilio Martin, Moises Martinez, Gustavo Recio, and Yago Saez. Pac-mant: Optimization based on ant colonies applied to developing an agent for ms. pac-man. Proceedings of the Symposium on Computational Intelligence and Games (CIG), 1:458 – 464, 2010.

[51] Risto Miikkulainen, Bobby D. Bryant, Ryan Cornelius, Igor V. Karpov, Kenneth O. Stanley, and Chern Han Yong. Computational intelligence in games. Computational Intelligence: Principles and Practice, 2006.

[52] Brad L. Miller and David E. Goldberg. Genetic algorithms, selection schemes and the varying effects of noise. Evolutionary Computation, 4, No. 2:113–131, 1996.

[53] Shahla Nemati, Mohammad Ehsan Basiri, Nasser Ghasem-Aghaee, and Mehdi Hosseinzadeh Aghdam. A novel aco–ga hybrid algorithm for feature selection in protein function prediction. Expert Systems with Applications, 36(10):12086 – 12094, 2009.

[54] Marcin Pilat and Tony White. Using genetic algorithms to optimize acs-tsp. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, Ant Algorithms, volume 2463 of Lecture Notes in Computer Science, pages 101–172. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-45724-0_28.

[55] Jean-Yves Potvin. A review of bio-inspired algorithms for vehicle routing jean-yves potvin. CIRRELT, 30(30), 2008.

[56] Timothy E. Revello and Robert McCartney. Generating war game strategies using a genetic algorithm. Proceedings of the Congress on Evolutionary Computation, 1:1086 – 1091, 2002.

[57] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph., 21:25–34, 1987.

[58] E. Ridge and E. Curry. A roadmap of nature-inspired systems research and development. Multiagent Grid Syst. 3:3–8, 2007.

[59] Xiaogang Ruan and Daoxiong Gong. A hybrid approach of ga and aco for tsp. In Proceedings of the 5'" World Congress on Intelligent Control and Automation, 2004.

[60] Noor Shaker, Julian Togelius, Georgios N. Yannakakis, Ben George Weber, Tomoyuki Shimizu, Tomonori Hashiyama, Nathan Sorenson, Philippe Pasquier, Peter A. Mawhorter, Glen Takahashi, Gillian Smith, and Robin Baumgarten. The 2010 mario ai championship: Level generation track. IEEE Trans. Comput. Intellig. and AI in Games, 3(4):332–347, 2011.

[61] Julian Togelius. Mario ai competition. In Lanzi [46].

[62] Francis C.M. Lau Xiang Feng and Daqi Gao. A new bio-inspired approach to the traveling salesman problem. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 5:1310–1321, 2009.

[63] Yi-Liang Xu, Meng-Hiot Lim, Yew-Soon Ong, and Jing Tang. A ga-aco-local search hybrid algorithm for solving quadratic assignment problem. In Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006.

[64] Yang and Xin-She. Artificial intelligence and knowledge engineering applications: A bioinspired approach. In LNCS, pages 317–323. Springer Berlin / Heidelberg, 2005.

[65] Michael Mascagni Yaohang Li. A bio-inspired job scheduling algorithm for monte carlo applications on a computational grid. In IMACS -WORLD CONGRESS, 2005.

[66] Alayed, H., Frangoudes, F., and Neuman, C. (2013). Behavioral-based cheating detection in online first person shooters using machine learning techniques. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

[67] Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43.

[68] Berns, A., Gonzalez-Pardo, A., and Camacho, D. (2013). Game-like language learning in 3-d virtual environments. *Computers and Education*, 60(1):210 – 220.

[69] Boreczky, J. S. and Wilcox, L. D. (1998). A hidden markov model framework for video segmentation using audio and image features. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 6, pages 3741–3744. IEEE.

[70] Brandstetter, M. F. and Ahmadi, S. (2012). Reactive control of ms. pac man using information retrieval based on genetic programming. In *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012, Granada, Spain, September 11-14, 2012*, pages 250–256.

[71] Brzezinski, D. (2010). Mining data streams with concept drift. Master's thesis, Poznan University of Technology.

[72] Buckley, J. and James, I. (1979). Linear regression with censored data. *Biometrika*, 66(3):429–436.

[73] Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. (2008). Monte-carlo tree search: A new framework for game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, October 22-24, 2008, Stanford, California, USA*.

[74] Dey, R. and Child, C. (2013). Ql-bt: Enhancing behaviour tree design and implementation with q-learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

[75] Dietterich, T. G. (2002). Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings*, volume 2396, pages 15–30.

[76] Drachen, A. and Canossa, A. (2009). Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, pages 202–209. ACM.

[77] Drachen, A., Canossa, A., and N., G. (2009). Player modeling using self-organization in tomb raider: Underworld. In *Proceedings of the 5th International Conference on Computational Intelligence and Games*, CIG'09, pages 1–8, Piscataway, NJ, USA. IEEE Press.

[78] Fan, W. (2012). Graph pattern matching revised for social network analysis. In *Proceedings of the 15th International Conference on Database Theory*, volume 548, pages 8–21. ACM.

[79] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

[80] Figueiredo, M. A. and Jain, A. K. (2002). Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396.

[81] Gagne, D. J. and Congdon, C. B. (2012). Fright: A flexible rule-based intelligent ghost team for ms. pac-man. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 273–280. IEEE.

[82] Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, volume 2, pages 42–47. Association for Computational Linguistics.

[83] Gonzalez-Pardo, A., Palero, F., and D., C. (2014a). An empirical study on collective intelligence algorithms for vide games problem-solving. *Computing and Informatics*, In press.

[84] Gonzalez-Pardo, A., Palero, F., and D., C. (2014b). Micro and macro lemmings simulations based on ants colonies. In *Evostar. EvoGames*, page In press.

[85] Gonzalez-Pardo, A., Rosa, A., and D., C. (2014c). Behaviour-based identification of student communities in virtual worlds. *Computer Science and Information Systems*, 11(1):195–213.

[86] Hastie, T. and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(6):607–616.

[87] Hearst, M. A. (1998). Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

[88] Johansson, A. and Dell'Acqua, P. (2012). Emotional behavior trees. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 355–362. IEEE.

[89] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

[90] Karakovskiy, S. and Togelius, J. (2012). The mario ai benchmark and competitions. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):55–67.

[91] Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., et al. (2001). Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679.

[92] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden markov models in computational biology: Applications to protein modeling. *Journal of molecular biology*, 235(5):1501–1531.

[93] Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots + machine learning. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 435–442.

[94] Liu, R. and Zhang, H. (2004). Segmentation of 3d meshes through spectral clustering. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 298–305. IEEE.

[95] Menéndez, H. D., Barrero, D. F., and Camacho, D. (2013a). A multi-objective genetic graph-based clustering algorithm with memory optimization. In *IEEE Congress on Evolutionary Computation (CEC 2013)*.

[96] Menéndez, H. D., Plaza, L., and Camacho, D. (2013b). A genetic graph-based clustering approach to biomedical summarization. In *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM.

[97] Nguyen, K. Q., Wang, Z., and Thawonmas, R. (2013). Potential flows for controlling scout units in starcraft. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–7. IEEE.

[98] Olson, J. R. and Olson, G. M. (1990). The growth of cognitive modeling in human-computer interaction since goms. *Human–computer interaction*, 5(2-3):221–265.

[99] Pennacchiotti, M. and Popescu, A.-M. (2011). A machine learning approach to twitter user classification. *ICWSM*, 11:281–288.

[100] Perez, D., Nicolau, M., O'Neill, M., and Brabazon, A. (2011). Evolving behaviour trees for the mario AI competition using grammatical evolution. In *Applications of Evolutionary Computation - EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Torino, Italy, April 27-29, 2011, Proceedings, Part I*, volume 6624, pages 123–132. Springer.

[101] Picard, R. W. (2003). Affective computing: challenges. *International Journal of Human-Computer Studies*, 59(1):55–64.

[102] Polceanu, M. (2013). Mirrorbot: Using human-inspired mirroring behavior to pass a turing test. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

[103] Powley, E. J., Whitehouse, D., and Cowling, P. I. (2012). Monte carlo tree search with macro-actions and heuristic route planning for the physical travelling salesman problem. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 234–241. IEEE.

[104] Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[105] Ray, S. and Turi, R. H. (1999). Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, pages 137–143.

[106] Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

[107] Romero, C. and Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert systems with applications*, 33(1):135–146.

[108] Rosenthal, C. and Congdon, C. B. (2012). Personality profiles for generating believable bot behaviors. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 124–131. IEEE.

[109] Safavian, S. R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.

[110] Schaul, T. (2013). A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

[111] Shaker, N., Togelius, J., Yannakakis, G. N., Weber, B., Shimizu, T., Hashiyama, T., Sorenson, N., Pasquier, P., Mawhorter, P., Takahashi, G., et al. (2011). The 2010 mario ai championship: Level generation track. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(4):332–347.

[112] Shaker, N., Yannakakis, G. N., and Togelius, J. (2010). Towards automatic personalized content generation for platform games. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010, October 11-13, 2010, Stanford, California, USA*.

[113] Sharma, K., Shrivastava, G., and Kumar, V. (2011). Web mining: Today and tomorrow. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 1, pages 399–403. IEEE.

[114] Sifa, R. and Bauckhage, C. (2013). Archetypical motion: Supervised game behavior learning with archetypal analysis. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE.

[115] Synnaeve, G., Bessiere, P., et al. (2011). A bayesian model for plan recognition in rts games applied to starcraft. In *AIIDE*.

[116] Traish, J. M. and Tulip, J. R. (2012). Towards adaptive online rts ai with neat. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 430–437. IEEE.

[117] Vinciarelli, A., Pantic, M., and Bourlard, H. (2009). Social signal processing: Survey of an emerging domain. *Image and Vision Computing*, 27(12):1743–1759.

[118] Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA. ACM.

[119] Weber, B. G. and Mateas, M. (2009). A data mining approach to strategy prediction. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milano, Italy, 7-10 September, 2009*, pages 140–147.

[120] Yannakakis, G. N. and Maragoudakis, M. (2005). Player modeling impact on player's entertainment in computer games. In *Proceedings of the 10th International Conference on User Modeling*, UM'05, pages 74–78, Berlin, Heidelberg. Springer-Verlag.

[121] Young, J., Smith, F., Atkinson, C., Poyner, K., and Chothia, T. (2012). Scail: An integrated starcraft ai system. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 438–445. IEEE.

[122] Zalik, K. R. (2008). An efficient k'-means clustering algorithm. *Pattern Recognition Letters*, 29(9):1385–1391.